DOCUMENT RESUME

ED 025 018 24 EA 001 817

Functional Analysis and Preliminary Specifications for a Single Integrated Central Computer System for Secondary Schools and Junior Colleges. A Feasibility and Preliminary Design Study. Interim Report.

Computation Planning, Inc., Bethesda, Md.

Spons Agency-Office of Education (DHEW), Washington, D.C. Bureau of Research.

Bureau No-BR-8-9011

Pub Date May 68

Contract-OEC-0-8-009011-2507

Note-71p.

EDRS Price MF-$0.50 HC-$3.65

Descriptors-*Computer Oriented Programs, Computer Storage Devices, Data Processing, Educational Benefits, Estimated Costs, *Feasibility Studies, Financial Support, Input Output Devices, *Junior Colleges, Programing, *Secondary Schools, *Specifications, Vocational Education

Identifiers-ALGOL, COBOL, FORTRAN

A feasibility analysis of a single integrated central computer system for secondary schools and junior colleges finds that a central computing facility capable of serving 50 schools with a total enrollment of 100,000 students is feasible at a cost of $18 per student per year. The recommended system is a multiprogrammed-batch operation. Preliminary specifications are given for the system's objectives, costs, capacity, flexibility, and growth capability and for alternative types of software, hardware, formats, and compilers. Benefits of the specified system are discussed with respect to educational values, vocational training values, and administrative applications. Twenty-one appendices analyze in detail aspects of the proposed system and present supporting information and data. (TT)

Phase I of Final Report
BR 8-9011
PA-24

COMPLAN t.m.

INTERIM REPORT

Functional Analysis and Preliminary Specifications
for
A SINGLE INTEGRATED CENTRAL COMPUTER SYSTEM
FOR
SECONDARY SCHOOLS AND JUNIOR COLLEGES

A Feasibility and Preliminary Design Study
Performed for the
U. S. Office of Education

under

Contract No. OEC-0-8-009011-2507

This Report Prepared May, 1968

by

COMPUTATION PLANNING, Inc.
7979 Old Georgetown Rd.
Bethesda, Maryland 20014

301-654-1800

ED025018

EA 001 817

COMPLAN$^{t.m.}$

INTERIM REPORT

Functional Analysis and Preliminary Specifications
for
A SINGLE INTEGRATED CENTRAL COMPUTER SYSTEM
FOR
SECONDARY SCHOOLS AND JUNIOR COLLEGES

A Feasibility and Preliminary Design Study
Performed for the
U. S. Office of Education

under

Contract No. OEC-0-8-009011-2507

This Report Prepared May, 1968

by

COMPUTATION PLANNING, Inc.
7979 Old Georgetown Rd.
Bethesda, Maryland 20014

301-654-1800

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
OFFICE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL OFFICE OF EDUCATION POSITION OR POLICY.

ACKNOWLEDGEMENT

This study was carried forward under the guidance and with the assistance of several senior staff members of the U.S. Office of Education, whose support and encouragement are gratefully acknowledged. We also acknowledge the cooperation we received on many occasions from the Project Managers for the two earlier studies to which this is a sequel.

TABLE OF CONTENTS

Page

1. INTRODUCTION AND SUMMARY: RECOMMENDATIONS 1

- 1.1 System 1
- 1.2 Software 1
- 1.3 Hardware 1
- 1.4 Operations 1

2. PROBLEM DEFINITION 1

- 2.1 Potential Values of Computation 1
  - 2.1.1 Educational Values 1
  - 2.1.2 Vocational Training Values 2
  - 2.1.3 Cautionary Note on Student Guidance 2
  - 2.1.4 Administrative Applications 2
- 2.2 Intended Capability 2
- 2.3 Problems of Penetration 3
- 2.4 A Recommended Strategy for Penetration 3

3. PRELIMINARY SPECIFICATIONS OF RECOMMENDED SYSTEM 4

- 3.1 General Discussion 4
  - 3.1.1 System Concept 4
  - 3.1.2 Objectives 4
  - 3.1.3 Justification of Decisions 5
    - 3.1.3.1 Relative Costs 5
    - 3.1.3.2 Capacity 5
    - 3.1.3.3 Flexibility 5
    - 3.1.3.4 Growth Capability 5
  - 3.1.4 Software 5
  - 3.1.5 Hardware 5
  - 3.1.6 Operations 5
- 3.2 Software 6
  - 3.2.1 General 6
  - 3.2.2 Multiprogrammed-Batch Executive System 6
    - 3.2.2.1 General Characteristics 6
    - 3.2.2.2 Job Sequencing 7
      - 3.2.2.2.1 Student Jobs 7
      - 3.2.2.2.2 Administrative Jobs 7
      - 3.2.2.2.3 Official Query 7
    - 3.2.2.3 Resource Allocation 7
      - 3.2.2.3.1 Main Memory Space 7
      - 3.2.2.3.2 Auxiliary Memory Space 7
      - 3.2.2.3.3 Processor Assignment 8
      - 3.2.2.3.4 Communications Control 8
      - 3.2.2.3.5 Space Scavenging 8
    - 3.2.2.4 Input-Output 8
      - 3.2.2.4.1 System Standard Formats 8
      - 3.2.2.4.2 User Specified Formats 8
  - 3.2.3 Compilers 8
    - 3.2.3.1 General Remarks 8
    - 3.2.3.2 FORTRAN 9
    - 3.2.3.3 ALGOL 9
    - 3.2.3.4 COBOL 9
    - 3.2.3.5 Other Procedural Language Translators 9
    - 3.2.3.6 Non-Procedural Language Handlers 9
    - 3.2.3.7 SYSTEM CONTROL LANGUAGE 9

Page
3.3 Hardware 9
3.3.1 General 9
3.3.2 Data Central 9
3.3.3 Remote Input/Output Stations 10
3.3.4 Communications 10
3.4 Operations 10

4. ESTIMATED COSTS FOR SYSTEM REPLICATION 10

4.1 Hardware 10
4.2 Software Development and Checkout 10
4.3 Space 10
4.4 Operations Personnel and Vehicles 10
4.5 Supplies 10
4.6 Purchase vs. Rental of Hardware 10
4.7 Enlarged School Population (200,000 Students) 11
4.7.1 Discussion 11
4.7.2 System Configuration for 200,000 Population 11
4.8 Overall Total Cost Per Student 11
4.9 Remarks on Comprehensiveness of Cost Considerations 11

5. SUMMARY OF SYSTEM PARAMETERS 12

5.1 Communications System Parameters (including Buffer Sizes) 12
5.2 Fast Memory Speed and Size 12
5.3 Auxiliary Memory Sizes, Access Times, and Transfer Rates 12
5.4 Central Processor Parameters and Utilization 12
5.5 Average Turnaround Time for Various Usage 12
5.7 Billing System 12
5.8 Manning 13
5.9 Financial Support Pattern 13

6. CONCLUSIONS ON FEASIBILITY 13

6.1 Educational 13
6.2 Economic 13
6.3 Operational 13

OUTLINE OF APPENDICES

Appendix	Section	Description
A	Title	Operating Modes: Batch vs. TimeSharing vs. "Quick-Batch"
	1.	Introduction
	2.	Definitions of Operating Modes
	2.1	Batch
	2.2	TimeSharing
	2.3	Quick-Batch
	3.	Student Support Considerations
	3.1	Response Time
	3.2	Interactivity
	3.3	System Support Capabilities
	3.4	Program Input Preparation: On-Line Key-In vs. Off-Line Card
	3.5	Student Program Storage: On-Line vs. External
	4.	Economy
	4.1	Central System
	4.2	Terminal Hardware
	4.3	Communications
	4.4	Consumable Supplies
	4.5	Manpower
B	Title	Comparison of MP Batch Systems of GLC and IBM
	1.	Introduction
	2.	Quotation from IBM Report (on MP Batch System)
	3.	Discussion of GLC MP Batch System
	4.	GLC Configuration Recommendation
C	Title	Comparison of T-S Systems of GLC and IBM
	1.	Objectives
	2.	Comparison of Workload Assumptions
	2.1	Student Workload
	2.2	Console Time per Session
	2.3	Program Size
	2.4	I/O Data Rates
	2.5	Overload Provisions
	3.	Hardware
	4.	Software
	5.	Costs
	6.	Overall Reconciliation of Console Recommendations
	7.	Comparison Tabulations
	7.1	Workload Summary
	7.2	Hardware Summary
D	Title	Multiprogramming System - Analytical Discussion
	1.	Introduction
	2.	Factors in Scheduling from Queue
	3.	Background on Utilization of System Resources
	4.	Graceful Degradation
	5.	Communications from Remote locations to Central Facility
	6.	System Planning for Multiprogramming and Multiprocessing
E	Title	Input/Output Station Hardware
	1.	Introduction - Concept: Batch
	2.	Input
	2.1	Input Media
	2.2	Preliminary Choice: Punched/Marked Cards
	2.3	Card Reader Types
	3.	Output
	3.1	Media

Appendix	Section	Description
	3.2	Initial Choice: Medium-Speed Alphanumeric Printer
	3.3	Printer Types
	3.4	Ink-Stream Printer
F	Title	Nature of Billing Problems
	1.	Batch-Mode
	2.	Time-Sharing Mode
	2.1	The RAND Corporation JOSS System
	2.2	End-of-month Type
G	Title	Communication Economics
	1.	Introduction
	2.	Time-Sharing Communication Costs (per GLC)
	3.	Multiprogrammed Batch Communication Costs
H	Title	Financial Support Considerations
	1.	Introduction
	2.	Government-Support Recommendations
	2.1	Planning and development of hardware/software
	2.2	System and user performance research activities
	2.3	Teacher training
	2.4	Government support for voluntary cooperative teacher activities
	2.5	Planning and procurement of broadly useful follow-on developments
	3.	State and Local Support Recommendations
	3.1	System Operations
	3.2	Installation
	3.3	Maintenance
	3.4	Consumable supplies
	3.5	Communications services
	4.	Operation of Central Facility by Commercial Firm
I	Title	Operations
	1.	Introduction
	2.	Data Central Operations
	3.	Remote (In-School) Station Operations
	4.	Remote Station Operations Manual
	4.1	Routine Equipment Procedures
	4.2	Abnormal or Emergency Procedures
	4.3	Staffing Procedures
	4.4	Supplies
	4.5	System Messages
	4.6	Operating Policies and Procedures
J	Title	Workload
	1.	Introduction
	2.	Anticipated Morning Queue
	3.	Scheduled Demands
	4.	Student Program Listings
	5.	Penetration
	6.	Administrative Workload
K	Title	System Cost Matrix
	1.	Hardware Costs
	2.	Software Costs
	3.	Space (office and Laboratory) Costs for Pilot System

Appendix	Section	Description
	4.	Operations Personnel and Vehicles
	5.	Supplies
	6.	Discussion of Recommended System Costs
L	Title	File Security Considerations
	1.	Introduction
	2.	Types of Access Control
	3.	Time-Lock
	4.	Password
	5.	Physical Removal Control
	6.	Recommendations
M	Title	Physical Samples of Printer Output and Marked/Punched Cards
N	Title	Preparation and Handling of Decks and Programs
	1.	Preparation of Decks
	1.1	Keypunching by Students
	1.2	Keypunching by Paid Staff
	1.3	Mark-Sense Cards
	1.4	Low Cost Manually Operated Card Punch
	1.5	Recommendations for Student Deck Preparation
	1.6	Administrative Keypunching
	2.	Handling of Student Programs
	2.1	Students' own Program Decks
	2.2	Operator Handling
	2.3	Error Form
	2.4	Student Program Security
	2.5	Student Card Stock
O	Title	On-Line Storage of Student Programs
	1.	Introduction
	2.	Magnitude of Storage Requirements
	3.	Discussion of Costs
	4.	Recommendations
P	Title	Discussion of Programming Languages for Student Use
	1.	Introduction
	2.	ALGOL
	3.	APL
	4.	Use of COBOL and FORTRAN
Q	Title	Cooperative Creation of Library Programs by Teachers
R	Title	Discussion of COBOL Language Usage for Business Oriented Students
S	Title	Discussion of a "Quick-Batch" Language Processor for Student Programs
T	Title	Recommended Configurations
U	Title	Compiler vs. Interpreter for Student Programs

1. INTRODUCTION AND SUMMARY: RECOMMENDATIONS

The present report is based in part on the results of earlier studies by General Learning Corporation (GLC) and International Business Machines Corporation (IBM) on behalf of the USOE.

The recommendations that follow are based upon COMPLAN's considered evaluation of the earlier recommendations; of additional data obtained from both contractors and from educators in several institutions; and of the experience of COMPLAN technical staff members in personal association with university-level computing facilities, gained as student user, as instructor, as system programmer, and as computing center director.

Recommendations:

1.1 System

For beginning-level student usage, greater value for time-sharing (as opposed to batch) has not been demonstrated, but greater cost appears to be certain. Consequently, the multiprogrammed-batch concept should be specified in requests for proposals from vendors.

The system must be capable of handling inputs and returning outputs from 57 or more remote locations.

1.2 Software

The executive system must be capable of concurrent execution of several unrelated tasks. It must include job accounting functions and a priority scheme for handling jobs of different urgency.

The standard compiler for student jobs must be high-speed and reside in main memory. The recommended compromise choice as a language for this compiler is USA Standard FORTRAN.

The System must include conventional FORTRAN and COBOL compilers, with ALGOL 60 or some ALGOL dialect also desirable, but not mandatory.

1.3 Hardware

The control processor must be capable of processing at least 0.5 million instructions per second. Fast memory must be about 1/8 million words or 1/2 million characters with a maximum full cycle time of 1 microsecond. One measure of hardware speed and software performance is that a 200-statement FORTRAN program must be compilable by the system into machine object code in less than one second.

Sufficient auxiliary memory must be provided to contain administrative files and input/output queue buffers. To accomplish this it is estimated that 1-2 million characters of fast auxiliary memory and 500 million characters of slower auxiliary memory will be needed. This auxiliary storage must be easily expandable in modules without the need for reprogramming or extensive hardware modification.

The central facility must have high speed card reader (s) and printer (s) to handle some portions of the administrative workload.

The remote locations will have medium speed card readers and printers primarily to process the student workload.

1.4 Operations

The central facility should be operated and managed by a profit-oriented contractor.

The remote facilities should be operated by individual schools. Student program decks should be prepared by the students (mark-sense or manual-punch-punched cards) (cf. Appendix N) except for COBOL decks, which should be punched by clerical workers.

There is a lack of hard data regarding student performance in use of hand-marked cards as input to a batch system. For this reason, a limited number of manually-operated card punches should be provided and the decision as to whether it may be necessary to provide additional keypunch support, at least for initial preparation of unusually large programs on cards, must be deferred until meaningful test data has been acquired. Experiments to acquire such data could and should be performed during academic year 1968-69. Administrative data preparation and creation of COBOL program decks should be done using conventional keypunching methods.

2. PROBLEM DEFINITION

2.1 Potential Values of Computation

2.1.1 Educational Values

A frequently stated belief is that if one really wants to learn a subject he should teach a course in it. The computer has added a new dimension. The story now is extended to the belief that if one wants to check on whether he understands a concept, he should try to write a computer program involving the concept. The challenge to both students and faculty alike of telling a computer how to solve a problem -- that is of writing a computer program -- and of getting the computer to solve the problem correctly, incorporates a number of desirable educational steps and objectives, regardless of the course and the subject matter.

The potential user must arrive at a clear understanding of the problem to be solved. While this sounds trivial, it is unfortunately often the case that computer users have had unsatisfactory experiences with computers because they hadn't taken the actions necessary to define their problems clearly.

A second requirement is that the computer user must organize his thinking in an orderly sequence of steps to get from what is given to the solution of the problem. This requirement offers an unusual opportunity to individualize instruction to meet student capabilities. Bright students can be assigned intellectually challenging problems while less talented students can be given more routine tasks.

Another aspect of learning to use the computer is that one must write the computer instructions perfectly in order to obtain the correct solution to a problem. This comes as a shock to many students who have rarely had to perform any task with 100% accuracy in their academic experiences.

Finally, of course, the students will see how the computer can be used to solve a wide variety of problems. (See Sections 2.3 and 2.4 for further discussion of this area.) Our society has become highly automated and it is important that students have an understanding of the things a computer can and cannot do.

For the teacher the computer will often offer the opportunity to simulate more realistic problems. The amount of data to be used can be larger and the numbers do not have to be "nice". Some assistance can be given to teachers in grading certain kinds of tests. Multiple-choice tests lend themselves especially readily to automated techniques, and item analysis programs can assist the faculty in improving the quality of test items.

2.1.2 Vocational Training Values

The burgeoning computer industry needs many people at various levels of education and skill. Capturing data for input to computers is still largely accomplished by means of key-strokes on some device such as a keypunch, teletype machine, etc. It may be possible to train keypunch operators in some selected high schools, perhaps including an advanced course in which the students actually punch the programs written by other students (see Appendix N for further discussion).

Production control -- i.e., handling of computer input and output -- is also needed in the computer industry. Since every participating school will have a remote card reader and printer and thus the need for some form of production control, some students can gain experience at a very elementary level in that aspect of computer facility operation.

While someone will be needed to put cards into the card readers, take them out, remove output from the printer, and associate card decks with printout, it is improbable that this will be sufficient to be considered suitable training and experience for a job as a computer operator after graduation. In the case of schools located near the central computer facility, it may be possible to acquire a broader range of experiences in a computer center.

The nature of the programming courses and experience that a secondary school student will get in programming is enough to give the student some limited insight into what a career as a programmer would be like, but will be insufficient for students to go directly from the high school to a programming job, even at the trainee level. Additional training, preferably at the college level for many applications, will be needed for those who want to follow programming careers.

2.1.3 Cautionary Note on Student Guidance

Some programmers who have not had college education have been notably successful in performing certain difficult kinds of programming work, with or without formal training in programming. In many cases, however, such workers relatively soon reach their learning (and earning) ceilings.

We should distinguish carefully between training (i.e., learning how to do something) and education (i.e., learning how to learn more).

We feel that programming training courses are useful in helping students to learn quickly how to use computers in straightforward fashion, but the courses in computing concepts and in the advanced disciplines underlying related topics in mathematics, engineering, physical science, and linguistics are basic to preparation for creative work in computing science and in many other fields of endeavor.

For this reason, we feel that it is critically important for teachers to avoid removing from the higher-education stream students who have strong potential for advanced study. Thus, teachers, in their zeal to help students become early high-wage earners, should be wary of urging academically-promising students into the vocational-training program.

2.1.4 Administrative Applications

The GLC report, Appendix 2, describes many potential administrative computer applications; that exposition will not be repeated here. (See, however, Sections 2.3 and 2.4 below for related consideration.)

A high degree of cooperation between participating schools during periods of peak administrative workloads (e.g., at mid- and end-of-semester grading cycles, scheduling of students for following year, end of payroll periods, etc.) will be mandatory if sucess is to be achieved in the automation of the administrative applications.

2.2 Intended Capability

The purpose of the Central Computing Facility plan is to provide low-cost computation services for use by secondary school and junior college students, and by the administrative operations of their school systems. More specifically, the central facility will provide support for the following:

a. Students taking programming courses, both educational-preparatory (i.e., academic) and vocational (i.e., terminal-study) in orientation;

b. Students and faculty performing calculations and other applications in various academic subjects; and

c. School administrators and faculty preparing schedules and reports, maintaining records and performing other tasks involving numeric or non-numeric information processing by computer.

2.3 Problems of Penetration

Table 6.6 on page 6-6 of the GLC report presents estimated values of the "penetration factor" (i.e., fraction of students who use the computing facility). Very little hard data was available for the preparation of the table, yet the table entries play a vital role in the calculation of estimates of the system workload.

It is recommended that further study be given to estimating values for Table 6.6. Consideration of typical course offerings for grades 9-12 and the probability of computer usage in them on a course-by-course basis, plus further discussions with faculty members, should permit a refinement of the table entries. This is essential prior to preparation of the specifications to be submitted to vendors with the request for a proposal.

The table entries of GLC's Table 6.6 must be considered as potential long range target values for penetration. Many factors will contribute to the final results.

The number of teachers in the typical high school who have had meaningful computing experience can be expected to be extremely small. It is unrealistic to assume that, merely because a computer is made available to the school, there will be a strong demand on the part of the faculty to learn about computers, to learn programming, or to learn how the computer can assist them in their teaching and grading. People are reluctant to change from old to new methods, and teachers are no exception. Teachers can be expected to be reluctant to use the computer in their teaching when they lack understanding and experience in its use.

Outside assistance will be needed in training the teachers, and in particular, in training those teachers who will teach the programming courses. There will be a shortage of textbooks suitable for teaching programming to high school students, since most programming texts tend to be mathematically oriented and require at least the basic algebra course for FORTRAN programming and some understanding of business applications for COBOL programming. There will be virtually no texts in any of the subjects that include use of the computer in support of teaching or include problems suitable for a computer solution. In summary, the approach to substantial penetration can be expected to be slow.

Penetration for administrative use of the computer can be expected to go somewhat more rapidly, but not without reluctance and possibly outright resistance on the part of some administrators. Here, too, outside assistance will be necessary for analysis and design of applications. (See Section 2.4 on strategy).

2.4 A Recommended Strategy for Penetration

The first step must obviously be a determination of which schools are to participate in this program. A mere announcement of the programs and an invitation to submit a request for inclusion may produce few volunteers. It is recommended that presentations be made in as many schools and colleges as possible in the area selected for the pilot study prior to inviting the schools and colleges to indicate their interest.

If the pilot program is to begin in the school year 1969-1970, then training of teachers must be done in the year 1968-1969, which means the presentations recommended above must be prepared and given in the very near future. September 1968 is probably the earliest possible time for the presentations with selection of schools and beginning of the teacher training program no later than January 1969.

After the presentations, schools should be invited to submit requests to participate. If the number of requests to participate exceeds 50, a choice among schools can be made. If the number is less than 50, it is recommended that additional selection, as necessary, be made in order to assure an initial student population of 100,000. This sample size is desirable in order to later validate the sucess of this pilot model.

For the first semester, possibly even for the entire first year, the only courses in which the computer will be used will be the programming courses. Teachers will be needed for these courses and student enrollment may be low. A particular effort should be made to include participation by outstanding secondary school students. In the colleges some attention should be given to encourage participation by the students and faculty in the schools of education so future generations of teachers will be computer-knowledgeable upon graduation.

It is probable that mathematics teachers, preferably selected from volunteers, will be the best choice to teach FORTRAN, and that teachers of business arithmetic and accounting should be selected to teach COBOL.

Other teachers who volunteer for programmer training should be accepted and trained and, if numbers permit, the principals can then select those who are to teach the programming courses.

The teacher training should include workshops on preparation of suitable teaching materials for the students and must include active programming opportunities. This means that some funds for the training program itself and for computer time should be included in the budget for FY 69.

If possible, the computer time should be contracted for with a local service bureau and the programs should be processed from remote facilities, thus simulating the planned operating environment of schools. It would be desirable, but is unlikely, than the same computing hardware, central and remote, that will eventually be used in the schools should be used for the training program. It is likely that appropriate software may be available; this should influence choice of vendor of the workshop support services.

The magnitude of the program to train the teachers should not be underestimated. If 50 schools elect to participate and send 4 teachers each, 200 teachers must be trained. Classes should be no larger than 25, because there needs to be appreciable interaction between the (teacher-) students and the programming instructor. This is especially necessary when programming errors are being identified and corrected. There is also the logistics problem of providing classroom space in a location.

In addition to the teachers selected for training in order to become prepared to teach the programming courses, teachers from other subject areas should also be included in the training program in preparation for penetration of computer usage into the curriculum in a variety of subject areas. It is recommended that just a few courses be selected for the initial efforts at penetration and that they be ones for which it is believed that there is a high probability of meaningful use of the computer.

There should be no hard and fast rule on which courses are selected; it may vary from school to school. Interest on the part of a faculty member in learning to use the computer in his teaching is of prime importance. It is anticipated that subjects in the physical sciences such as mathematics and physics, and subjects in the commercial area that involve calculations, such as business arithmetic and accounting, are the most promising candidates. We feel that it will be better to undersell use of the computer and to penetrate slowly, with positive results, than to oversell it and essentially force it on reluctant faculty members.

The faculty who receive programming training in the year 1968-1969 may be ready for some fledging experience in use of the computer in their teaching by the second semester of the year 1969-1970, or possibly late in the first semester of that year after some students have begun the programming courses.

Penetration of computer applications in meeting the administrative requirements of the school may proceed more rapidly than in the student-usage areas. Many schools will already have automated some portion of their administrative work. It is recommended that the first step be to ascertain what administrative applications are already automated and to attempt to get all schools automated to about the same starting base.

This should make possible the analysis and design of some general purpose programs, probably done on a contract basis with a reliable software house, that can handle everyone's requirements. Some schools have already had negative experiences in using computers for administrative work because of automating poorly chosen applications, being promised unattainable results by smooth-talking salesmen, being given poorly designed systems, and/or neglecting human factors in planning for automation.

Particular care should be given to the human factor considerations. Capturing data and putting it into machine-readable form most often depend on humans. The benefits to be gained from automating an application must clearly outweigh any additional work or inconvenience created or the humans involved will react negatively. Too often human factors have been improperly evaluated or neglected entirely with disastrous results.

As is the case with the student-usage application of computing, it will be better to penetrate the administrative applications slowly with possitive results than to move too rapidly with negative results. Periodic meetings of the appropriate administrative staff members to discuss problems encountered in systems already automated and to clarify requirements for new systems should enable the administrative applications area to move steadily ahead.

The existence of potentially-valuable highly-generalized and comprehensive systems of school administration software facilities, as have been developed by Project OTIS and NEEDS, should be publicized to school administrators and teachers, and seminars should be held to examine, in semi-public forum, positive and negative considerations in the decision as to whether such a system should be adopted by this facility and its participating schools.

3. PRELIMINARY SPECIFICATIONS OF RECOMMENDED SYSTEM

3.1 General Discussion

3.1.1 System Concept

In considering earlier studies of both "Time-Sharing" (i.e., interactive usage from remote typewriter-like terminals) and "Batch" (i.e., single-request, single-response usage), we have concluded that:

(a) Time-sharing usage's pedagogical effectiveness is believed by many people to be greater than that of batch usage, particularly for advanced-level students; however, little conclusive evidence exists to support this belief at most levels, and essentially no such evidence exists with respect to usage by beginning-level students.

(b) Time-sharing usage workload requirements cannot be determined from earlier observations in educational environments with a comparable level of meaningfulness to that with which batch usage workload requirements can be determined from observations. This is primarily because of the smaller amount of valid experience to date with time-sharing, and because the uncertainties with respect to relative effectiveness make invalid the derivation of time-sharing requirements from batch experience.

(c) It appears that time-sharing usage workload requirements are not only less accurately determinable than those of batch usage, but that they are substantially larger expressed in terms of both central system support cost and terminal hardware cost if beginning-level student usage requirements are to be supported.

3.1.2 Objectives

The quick batch system recommended must be designed to optimize processing of large number of small student jobs. The basic concept is to have a central facility of suitable capacity to service a minimum of 57 remote facilities plus the capability to introduce jobs which require large amounts of input and/or output at the central facility. To handle the large volume of input/output from the remote facilities it is believed possible that the system design needs a communications processor.

The primary mode of operation of the system will be multiprogramming, a system operating technique that provides for concurrent execution of several unrelated tasks.

3.1.3 Justification of Decision

3.1.3.1 Relative Costs

In examining costs for the multiprogramming-batch system we noted that a large fraction of total system cost is attributable to the terminals and communications equipment. We believe that significant savings can be made by using a new class of medium-performance, low-cost card reader and printer equipments*. At least a few kinds of such hardware have become commercially available at announced prices. The new equipment also permits savings in communications costs.

Thanks to these economies, the total equipment cost (including all items covered in GLC's "total" cost estimate of $22 per student year) is $12 or $16 per student year, for purchased or leased Data Central hardware respectively; details are shown in Appendix K.

We believe the GLC estimates for time-sharing requirements to be fairly realistic and have therefore not made further detailed analysis of time-sharing costs.

We estimate additional costs of $6 per student year for the multiprogramming-batch system for facility preparation and operation, operating and support personnel, maintenance of purchased equipment, software maintenance, consumable supplies, courier service and administration.

Similar costs for time-sharing may be expected to be somewhat higher, since additional facility preparation will be required and trained adult personnel will be needed to monitor the terminal facilities and to provide assistance to users.

The ratio of the total costs for a time-sharing system to that for a multiprogramming-batch system is thus crudely estimated to be 2 to 2.5:1.

3.1.3.2 Capacity

Based on the data in the earlier reports it has been estimated that 57 remote terminals will be needed for the workload. This provides for 1 remote terminal in all participating schools and 2 in some of the larger ones. It assumes that the average student will submit 6.6 programs for processing in a school year, and includes provision for handling the administrative workload of the schools.

3.1.3.3 Flexibility

Availability of card readers at the remote locations will provide for a highly flexible scheduling of the workload well adapted to the processing of short student jobs and handling of administrative queries. Longer administrative tasks will be sent to the central facility via courier for processing and return.

3.1.3.4. Growth Capability

System design which includes a communications processor for handling of input/output functions is readily adaptable to growth by the addition of more remote facilities as the need arises.

Design of an executive system for multi-programming must be highly flexible to handle the dynamic changes in the work situation. This in turn implies an accompanying flexibility in adaptation to growth in the hardware resources of the system needed to accommodate an increased workload. It is within the state-of-the-art today to provide for such growth without extensive changes in existing programs.

3.1.4 Software

Software recommended includes:

(a) An executive system capable of operating in a multiprogramming mode, probably designed expecially for this application. It must include a priority handling feature and job accounting functions

(b) A fast compiler for student jobs. This compiler must be resident in main core and will probably be for USASI FORTRAN.

(c) Conventional FORTRAN, COBOL, and ALGOL compilers

(d) General utility routines and report generator (s)

(e) A special applications package, including such things as a linear programming capability, statistical routines, and matrix handling programs

3.1.5. Hardware

The central computer facility will include one or more central processors, high speed main memory, a large amount of auxiliary memory to be used mainly for administrative files, a communications processor to handle input/output functions, magnetic tape units, and several high speed card readers and printers.

Conventional commercially available communications lines will link the central facility to the remote locations.

Low cost card readers and printers at the remote locations will process the input and output. Several schools with larger workloads will have more than one card reader and printer.

3.1.6. Operations

Almost all of the student workload (other than student business data processing training course programs written mainly in COBOL) will be processed through the card readers and printers at the remote locations.

---

* Discussed in Appendix E.

Some portions of the administrative workload, for which only limited input/output is required (e.g., administrative queries) will be entered into the system from the remote locations. Other protions of the administrative workload and student COBOL jobs will be sent to and from the central facility via a courier. Most administrative jobs will be processed after the end of the school day.

The central facility will be managed and operated by a contractor. It will probably be necessary to guarantee the contractor some minimum monthly commitment and permit him to sell time to outside users on second or third shift.

The equipment and operations at the remote locations will be the responsibility of the participating schools, including the responsibility for the communications from their location to the central facility. People to operate the equipment and handle the production control at the remote locations will come from student volunteers for the student workload and school employees for the administrative workload.

3.2 Software

3.2.1. General

Since the bulk of the jobs to be processed will be student programs, the overall design should be oriented to small jobs. As such, the system should have a fast compiler* which does not spend a lot of time optimizing object programs, since most student programs will be executed only a few times and will have very short execution times.

Compile diagnostics for user source program syntax errors must be as clear and explicit as possible to assist the students in self-debugging of programs and thus reduce some of the "consultant" workload on the teachers of the programming courses.

Multiprogramming will be the general mode of operation with priorty system favoring jobs with a short run time and low output volume.

In addition to the fast compilers needed for student work, conventional compilers which do attempt to optimize object code should be provided by the vendor as part of the standard software package. These will be used for the compilation of programs constituting the administrative workload and other applications which require more efficient object code.

A simplified query capability is necessary to handle official queries as part of the administrative workload. The executive system must recognize a query and the priority it has been assigned, and must respond with service commensurate with the priority assigned.

3.2.2. Multiprogramming-Batch Executive System

3.2.2.1. General Characteristics

The executive system must:

(1) Be capable of handling in a multiprogramming mode 2 or more compilations, 57 or more inputs from card readers to buffer areas, and 57 or more outputs from the buffer areas to remote printers. The latter two types of jobs should probably be controlled by a communications processor designed to handle input/output requirements of the system. The basic design of the executive must, moreover, provide for at least 114 remote stations in order to allow ready expansion to serve 200,000 students in 100 schools.

(2) Move from job to job without operator intervention and must be tolerant of student errors. In particular, the system must be designed so that it does not "halt" when it encounters such things as errors in control card set-up, invalid characters on cards and similar blunders.

(3) Be readily modified. Each program presented for processing must be handled under executive control with regard to loading, sequencing, priority in the job queue, input/output and library calls, resource allocation and program termination. Jobs presented for processing may be random mix of FORTRAN, ALGOL, AND COBOL compilations, official queries, command language instructions and execution of previously compiled programs or "compile-and-go" programs.

(4) Include job accounting functions for billing purposes and later analysis of the utilization of system reaources.

(5) Be capable of adjusting to changes in system resources without the need for reprogramming either the system software or the applications programs.

In selecting jobs for processing, the executive system must respond to a priority system. In general priority 1 jobs should be processed before priority 2 jobs, priority 2 before priority 3, etc.

---

* Discussed in Appendix S.

3.2.2.2. Job Sequencing

3.2.2.2.1 Student Jobs

Student jobs will have a priority indicated on the header card, and will also have as mandatory entries limits on execution time and output volume. The jobs will have one of two priorities which may be described as "student-waiting" and "student-deferred". "Student-waiting" may be assigned at student request for student jobs with both short execution time and low output volume. The value of the parameter settings must be readily adjusted, but for example might be 1 second execution time and 300 lines of output. On a priority range of 1-5, student-waiting jobs would be authorized priority 2 and the student deferred jobs, priority 3. The latter group would consist of student programs for which the student did not wish to wait for results, and jobs for which the execution time or volume or output exceeded the established limits. For student jobs exceeding the established limits, it is recommended that faculty approval must be obtained in advance and that the header card include the name of the faculty member. As part of the job accounting analysis procedures a list should be produced, sorted by faculty member, listing all header cards on which specified execution time or output volumes exceeded the established limits. This will provide sufficient control to detect students requesting longer runs or more voluminous output and at the same time not increase the complexity of the executive system job accounting functions, since the analysis procedure would be a separate program which uses the job account file as its input data.

Because of state-of-the-art software limitations on COBOL, COBOL compilations and executions will be assigned priority 4.[§1]

An interesting alternative which would permit a user to specify the desired time his output should be ready for pick-up was considered. It is expected that most students would request pick-up at either lunch hour or the end of the school day. This would force the system to produce lead time queues in order to meet requests which would cause retention of sizable amounts of output for later printing.

The increased complexity of designing the executive to accomplish this function and associated costs in carrying it out are felt to exceed any benefits that might be gained.

3.2.2.2.2 Administrative Jobs

Administrative jobs should always have a priority indicator on the header card. Jobs which require immediate processing, e.g., daily attendance reports, should be assigned priority 1 (on the same 1-5 range). Most administrative jobs should not be submitted for processing until after the end of the school day. If they are submitted earlier and are not of the immediate type, they should be assigned priority 5.

3.2.2.2.3 Official Query

Some jobs of this type will require an almost immediate response, e.g., a guidance counsellor needing information during a student interview, and will need a priority 1 assigned. Other queries which do not have the same degree of urgency should be assigned a lower priority. All official queries should be assigned a priority on the header card.

3.2.2.3 Resource Allocation

3.2.2.3.1 Main Memory Space

Some of the problems in memory allocation have been related to excessive system overhead resulting from operations associated with multiprogramming and dynamic memory allocation schemes. A modified form of dynamic memory allocation is therefore proposed as follows: for jobs identified as priority 2 student jobs, a few fixed areas of memeory should be reserved. Limited multiprogramming of compilations and executions of these jobs should be permitted, the exact number to be determined after some operating experience with the system. Compilations and/or executions of the priority 2 student jobs should automatically be assigned one of the reserved areas previously mentioned. If, upon trying to assign memory space in the reserved area, the system recognizes that the job won't fit, then the system should allocate memory space outside the reserved area so the job can be processed, but the system should also suspend initiation of other jobs until that job which required the extra space is terminated. During the early days of the pilot system operation, some memory maps should be taken when student jobs are being processed to give empirical data on how large the areas reserved for the student jobs should be. The executive system should be designed so that the operator at the central facility can control the multiprogramming characteristics of the system. After some empirical data of the trade-off between multiprogramming and system overhead have been collected and analyzed it can be determined what an optimum job mix should be.

Memory allocation control should be by the (now-conventional) linked-list technique in order to permit single-priority queues to be serviced without checking of other queues.

3.2.2.3.2 Auxiliary Memory Space

There are two clases of requirements that call for auxiliary memory. The first may be classified as rapid-demand functions and include such things as input/output queue buffering or moving program segments and data. These requirements might be satisfied by some form of high performance auxiliary memory such as a fast drum or bulk core memory.

The second class of requirements for auxiliary memory is not so dependent upon high performance characteristics. Typical of this class are such things as administrative files and the system library. A form of mass memory such as a disc file will usually satisfy these requirements.

[§1] cf. Appendix R.

The executive system should assign both forms of auxiliary memory dynamically, but with low system overhead costs. It should be possible for the hardware to be configured with smaller amounts of auxiliary memory during the first year or two of operation of the pilot system, and then to add memory in modular increments as is required. The executive system must be capable of utilizing the memory added in modular increments in an efficient manner without the necessity of reprogramming either the applications programs or the executive system itself.

3.2.2.3.3 Processor Assignment

The system will operate in a multiprogramming mode, possible with multiple processors if such should be proposed by the successful bidder. In order to handle the workload, the processor(s) as a system resource must be shared by the various jobs in the mix. That is, if job A releases the processor in order to do an input/output operation, the executive system must be able to allocate this processor to job B which was in a queue waiting for a processor. The danger, of course, is that excessive system overhead in allocating system resources in an anonymous manner will offset gains made by virtue of multiprogramming. For this reason, the operator at the central facility must be able to control the extent to which the system engages in multiprogramming.

Other system resources for which there is redundancy should also be capable of being assigned by the executive in an anonymous fashion. This should permit more efficient use of the system resources and is in keeping with the recommended modularity of the system.

3.2.2.3.4 Communications Control

With 57 or more remote card readers and printers, in addition to higher-performance central site I/O hardware, it is necessary to handle communications in an efficient manner if the desired high rate of throughput of the system is to be realized. It is our belief that this can best be accomplished by providing a separate communications processor to handle the input/output requirements of the system. This communications processor will have access to the fast auxiliary memory. The executive system must place output or seek input in the buffer areas in the auxiliary memory which is shared with the communications processor.

3.2.2.3.5 Space Scavenging

Some systems attempt to coalesce programs in core to avoid the problem of "checkerboarding" of empty space in the core memory. This involves moving segments of programs and/or data. Obviously there is some overhead cost associated with such a process, although today's systems permit this to be done at an acceptable level of cost. In view of the recommendation for assigning some fixed memory areas for processing of student jobs, it is not felt necessary to move other programs to coalesce them and concurrently to coalesce available core memory space. Instead some form of linked list and dynamic memory allocation appears to be appropriate. To minimize system overhead, relocation hardware should be in the hardware design and the executive system should use the associated commands to accomplish overlaying efficiently, thereby minimizing the desirability of space scavenging.

3.2.2.4 Input/output

3.2.2.4.1 System Standard Formats

Since a large portion of the users of the systems will be students, the executive system design should be as tolerant and flexible as possible in accepting inputs and producing printed output. For input, the feature of free form on header and control cards and data cards should be a standard. For output, there should be a method of obtaining a standard output format by default if the user elects to omit format specifications.

3.2.2.4.2 User Specified Formats

The compiler for the bulk of the student programs must include a simplified format notation capability to avoid excessive debugging runs by novice users. Examples of the class of notation that might be approriate are:

```
In the source program:
      READ (A,B,C,)
On the data card:
      A=5.1, B=12, C=.0001756
In the source program:
      PRINT (FORCE, WEIGHT)
```

The output expected would then assume some standard format partly dependent upon the size of the value of the variable at run time. For the above, the output might be given the form:

```
FORCE=12.6
WEIGHT=1.234E+2
```

3.2.3 Compilers

3.2.3.1 General Remarks

For processing the students workload, the following characteristics of the compilers are needed:

(a) Good diagnostics, both at compilation time and execution time

(b) Fast compilation speeds, on the order of 1 second for 200 FORTRAN statements, with less emphasis on optimization of object code

All compilers must operate under the control of the executive system. The high-speed compiler for small student jobs must be resident in main memory. It is desirable that other compilers be re-entrant or Common Routines. Frequently used programs must be capable of being compiled to the system library for later call-out and execution. Maintenance of the system library must be an efficient and easily accomplished task.

The core resident compiler is to be FORTRAN. No other formula translating compiler approaches FORTRAN'S simplicity and high incidence-of-use within the United States.

3.2.3.2 FORTRAN

The resident FORTRAN compiler should include the entire set of capabilities as defined in the USA Standard FORTRAN (USASI -X3.9(1966)). The high speed version, resident in main memory for compilation of student jobs, must be "student-proof", that is, no student error, however bad, should be capable of destroying system software or causing system halts.

See Appendix S for a discussion of the WATFOR compiler which is representative of the type needed for the students programs in the secondary school environment.

The occasional need for handling larger FORTRAN compilations, such as library programs for teacher demonstrations and advanced student use, will require that a conventional FORTRAN compiler be provided in addition to the resident (small-job, high-speed) FORTRAN compiler.

3.2.3.3 ALGOL

While ALGOL has many desirable characteristics, there have been few acceptable implementations of it. As a result, it is not widely used in the United States.

The availability of an ALGOL compiler with characteristics similar to WATFOR should be considered a merit factor in any proposals received from vendors. However, no vendor should be eliminated from competition on the basis of lack of availability of a manufacture-supported version of ALGOL or one of its dialects such as MAD, JOVIAL, NELIAC, etc.

3.2.3.4 COBOL

The COBOL capabilities provided by the computer should conform to the current USASI recommendations for USA Standard COBOL.

For student use, the availability of COBOL is an absolute requirement. However, because COBOL operates as a series of overlays; its compilations are quite slow. That is, the execution of a COBOL compilation will take much longer than that of a comparable core resident FORTRAN compilation.

For administrative applications a COBOL must be available. This compiler should be designed to do some optimization of the object code generated.

3.2.3.5 Other Procedural Language Translators

A simple language processor is needed to handle administrative queries accessing student files. Further study of the nature of the probable queries is needed before specifying characteristics of the query language. However, since most persons who will need to make administrative queries will not be computer oriented, some general characteristics of the query language can be identified:

(a) The language should be as nearly like normal English usage as possible
(b) The input format should be free form

The query capability must be designed in a flexible and modular manner to permit changes and additions with a minimum of modification to the query master program.

3.2.3.6 Non-Procedural Language Handlers

Problem description languages such as DYANA (a mechanical system describing scheme) and several kinds of Report Generators will be useful but their processors will be handled as library programs rather than as production compilers.

3.2.3.7 System Control Language

Some form of command language must be available to the operator at the central facility as well as operators at the remote locations. This is necessary in order to assist the central facility operator in keeping track of system resources, communicating with operators at remote locations, suspending processing or changing priorities of jobs, and similar functions. At the central facility the operator will ordinarily use the control console typewriter for input. At the remote locations the only possiblity, other than a phone call, is to provide for input of messages in the command language through the card reader. There must be adequate documentation for the student operators on how to use the command language, and the executive system must permit additions to be made to the command language in an easily implemented manner.

3.3 Hardware

3.3.1 General

The hardware selected for the proposed system will be composed of commercially available equipment, consisting of a high-powered computer and related hardware at Data Central (either leased or purchased) and medium speed card readers and printers (purchased) in the school (remote) stations. Appendix T shows the hardware configurations recommended for 100,000 and 200,000 students.

3.3.2 Data Central

Data Central will be equipped with computing hardware needed to meet the demands on the system. Since continued studies are required§1 in the area of workload requirements, it is not feasible to determine precisely the central facility equipment requirement at this time. The computer will be a high speed device with approximately 1/2 million characters or 1/8 million words of main (directly-addressable high-speed) memory. Both high and low speed secondary storage will be provided for Input/Output (I/O), library routines, etc. Four magnetic tape drives are to be provided in addition to one card read/punch and one high speed printer. (Tape usage is incidental).

---

§1 cf. Appendix J.

3.3.3 Remote I/O Stations

Remote I/O Stations for the proposed system will be equipped with new generation equipment.§2 Card readers capable of reading intermixed mark-sensed and punched information at the rate of 200-250 40-column cards per minute are available and appear to be economically (unit cost - $4000) and operationally advantageous.

A new class of printers, also, is now available, combining medium speed (over 100 characters per second) and relatively low unit cost (-$5000).

Both these readers and printers are now commercially available.

3.3.4 Communications

The proposed communication system configuration is simple. One dedicated half-duplex line will be required for each card reader and one for each printer. Since these equipments are designed for remote terminal applications, they are available with interfaces designed for the 202D data sets. A total of four data sets will be required for each station connected to the system, one for each device at the central facility and the same at each remote station. These data sets are to be rental items.

3.4 Operations §3

System operation procedures for the proposed multiprogramming-batch system are to be kept simple. The central facility is to be operated by paid employees who will have little or no need or opportunity for intervention in the processing of student programs. These same paid employees will, however, have primary responsibility for the processing of the larger administrative tasks such as scheduling and report card generation.

Remote station operation will involve selected students who will work voluntarily. These voluntary students will perform all of the necessary remote station processing functions.

In-school facilities will be operated solely by voluntary student personnel. A minimum of two students will man each station at all times. These students will be knowledgeable of all handling procedures required for student programs and administrative queries. Student programs should be submitted to the operators who will check the decks for proper control cards, etc. Program decks will then be processed through the card reader; each output listing will be attached to the proper deck when received and both will be returned to the student upon request.

4. ESTIMATED COSTS FOR SYSTEM REPLICATION

Appendix K details costs for the 100,000 student Pilot System and also for replicating the Pilot System after it has demonstrated feasibility. The 100K student replica represents a so-called Chinese copy of the Pilot System; the 200K student replica represents the basic Pilot System expanded to handle 200,000 students at 100 schools. The 200K student system is discussed in 4.7 below.

There will be some costs for pre-planning, including such things as analysis and evaluation of the data collected from the pilot system and used to determine changes in hardware, software, and operations, in planning for the replication of the system. Costs for a training program for staff and faculty of the schools to be involved in the replicated system must also be considered. These costs, which we estimate at $150,000, are not included in the data presented in Appendix K.

4.1 Hardware

Costs are expected to be approximately the same as for the pilot system (See Appendix K).

4.2 Software Development and Checkout

Although the replication will be quite similar to the pilot system, it is anticipated that some basic changes in the system design will be made as a result of the experience gained on the pilot system, development of additional administrative applications for the replicated system, and changes to take advantage of advances in the state-of-the-art.

Some reduction in software maintenance cost is expected since the rate of changes to the system should be less than for the pilot system.

4.3 Space

This should be the same as for the pilot system.

4.4 Operations Personnel and Vehicles

Costs should be the same as for Pilot System

4.5 Supplies

Same as for Pilot System

4.6 Purchase vs Rental of Hardware (cf. Appendix K, **1 and 8.)

Purchase of the central facility and amortization of its cost over a five year period offers another option which tends to decrease the student-year costs. However, present advances in the state-of-the-art continue at a rapid rate. Significant advances in bulk memory might make it desirable to make significant changes in the basic system design, which might be impossible if the system were purchased.

---

§2 cf. Appendix J.

§3 cf. Appendix I.

Equivalent yearly cost for purchased equipment was determined by the straight-line method, amortizing over 5 years, with interest cost at 8% on the declining balance.

The purchase-cost and lease-cost calculations are based on rough approximation of expected (as of 1970) commercial prices for the purchase and rental of appropriate-performance equipment, without maintenance service. (Maintenance by manufacture-supplied contract services has been priced separately and is included in annual costs for both acquistion schemes.)

4.7 Enlarged School Population (200,000 Students)

4.7.1 Discussion

A possible alternative which would reduce the cost per student per year would be to include more schools in the area served by the pilot system. In view of the lack of hard data on student usage and the expectation that penetration will be gradual over a period of several years, it is probable that this can be attempted without an appreciable increase in equipment at the central facility.

However, doubling the student population will not halve the per student cost per year since some costs will also double, such as those for terminal equipment, communication, personnel at remote locations, etc. The following summary represents a central facility to support 100 schools in a 150 mile radius with a student population of 200,000.

It is assumed that the number of schools of each size class is doubled from that of the 50-school profile but that the average length of leased communications lines remains the same at 12 miles. The number of reader/printer stations doubles (from 57 to 114).

The increased costs at the central and remote facilities are estimated in Appendix K based in the assumption that the 200K student system represents an expansion of the 100K student pilot system. In order to determine costs for a 200K student pilot system (not shown in Appendix K), it is only necessary to substitute the pilot system software costs shown in Appendix K for those of the 200K student replica system, which will result in a net increase of $202,000 per year; all other cost remain the same as shown for the 200K student replica.

Note that doubling population and more than doubling geographical area causes some costs (e.g., software development) to change not at all; some costs to increase only slightly (e.g., Data Central hardware costs increase 13%); and some costs to increase substantially (e.g., remote station hardware and communication costs double).

4.7.2 System Configuration for 200,000 Population is shown in Appendix T.

4.8 Overall Total Cost Per Student (For Replica Systems)

For the basic 100,000 student population, for systems replicated from the pilot system, costs per student per year were:

Hardware	$15.88	if D.C.* harware leased, or $12.88 if purchased.
Software	.74	
Space	.43	
Operations	1.45	
Supplies	.82	
Total	$19.32	per student year, DC hdwe. leased
or	$16.32	per student year, DC hdwe. purchased

For the enlarged population (200,000 students, 100 schools, 100 mile radius), costs per student per year for replicated systems were:

Hardware	$10.00	if D.C. hardware leased, or $8.93 if purchased.
Software	.47	
Space	.25	
Operations	.88	
Supplies	.81	
Total	$12.41	per student year, DC hdwe. leased.
or	$11.34	per student year, DC hdwe. purchased

Let us now summarize student costs for the two acquisition options and for the two population sizes:

Student Population Size	Grand Total Cost per Student Year D.C. Hdwe Leased	 D.C. Hdwe Purchased
100,000	$19.32	$16.32
200,000	$12.41	$11.34

4.9 Remark on Comprehensiveness of Cost Considerations

It should be noted here that the cost arithmetic in this report includes many elements that are often omitted from planning documents of this kind. COMPLAN feels, however, that these items should be included if only to draw to the attention of potential readers the fact that these items should not be neglected in early planning. These considerations are stipulated and discussed in Appendix K.

Because it seemed inappropriate in a system planning report, and because no meaningful basis for estimate was available to us, we have omitted any consideration of possible implications of the proposed system on requirements for increased faculty salary costs.

Likewise it did not seem appropriate to include certain costs directly related to the establishment and operation of the remote sites, in particular remote site preparation and keypunch operator salaries. This decision was justified on the following basis:

These costs are expected to be highly variable because of the anticipated availability of existing facilities and personnel in some of the schools.**

---

* "D.C." means Data Central.

** See also last two paragraphs of Appendix K.

time (if in excess of pre-established parameters), auxiliary memory (file) space and time, and input and output volumes.

5.8 Manning

Central Facility	
Manager	1
Programming staff	4
Operations	14(for 2 shift operation)
Clerical & support	3
Remote stations	
Volunteer students	1200
Hardware Maintenance	
Technicians	4
Courier Service	
Drivers	2

5.9 Financial Support Pattern

We recommend Federal Government support for overall planning and development, for system and user performance research, teacher-training, intra- and inter-regional cooperative development of broadly usable programs, and for procurement of broadly useful follow-on developments; local funding (including State) should be used for operations, installation, maintenance, consumable supplies, and communications.

6. CONCLUSIONS ON FEASIBILITY

It is concluded, as a result of this study, that the proposed central computing facility, for providing service to 50 schools with a total enrollment of 100,000 students, is feasible provided the recommended system concept (multi-programmed-batch operation) is used.

6.1 Educational

The recommended system will, in accordance with projected workload determinations provided to COMPLAN by the U.S. Office of Education, meet the educational support requirements prescribed in its work statement by USOE.

The operating pattern follows that which has seen widest use in educational institutions in this country, except that certain impediments to effective student learning--including, especially, that of excessive and unpredictable delay in delivery of the results of student usage of computing facilities--are to be abated or removed.

It is believed that the resulting system should be highly effective as an educational and training tool.

6.2 Economic

A primary goal of the study has been to achieve a dramatic improvement in user-support economy for typical student problems of small size, as compared with typical costs that would be observed in present-day systems designed for commercial or industrial-professional users.

We believe that the student support costs to be expected from the proposed system--$13 per student year for system support (as defined in the General Learning Corporation report to which the present report is a sequel) or $18 per student year for total costs including related services as defined in the present report--represent a significant improvement over present-day costs for comparable services.

Further improvement could be achieved by doubling the size of the student population to 200,000. Corresponding costs would be about $9 per student year for system support (per GLC) alone, or $12 for all costs.

Leasing the major computing hardware at the central computing site would add $3 per student year to costs for the basic 100,000-student population, or $1.70 per student year for the 200,000-student population system.

6.3 Operational

The proposed system is expected to meet the basic performance requirement for an average of 5 minutes' delay in computing service for students.

OUTLINE, APPENDIX A

Discussion of System Operating Modes: Batch vs. Time-Sharing vs. "Quick-Batch"

1. Introduction
2. Definitions of Operating Modes
    - 2.1 Batch
    - 2.2 Time-Sharing (Conversational)
    - 2.3 "Quick Batch"
3. Student Support Considerations
    - 3.1 Response Time
    - 3.2 Interactivity
    - 3.3 System Support Capabilities
    - 3.4 Program Input Preparation: On-Line Key-In vs. Off-Line Card Preparation
    - 3.5 Student Program Storage: On-Line vs. External
4. Economy
    - 4.1 Central System
    - 4.2 Terminal Hardware
    - 4.3 Communications
    - 4.4 Consumable Supplies
    - 4.5 Manpower

1. Introduction

This Appendix will introduce and compare the concepts of so-called "batch", "quick batch", and "time-shared" use of high-powered computing facilities, in the context of the student-support system environment.

The workload in such an environment is characterized by a large number of problems of small size. Most users perform elementary tasks consisting largely of self-contained programs, with some use of elementary library program facilities. Execution times of programs will typically be measured in seconds or fractions of a second if performed on a high-speed machine.

We will consider here the ways in which the system may be organized and used in order to support such a workload.

2. Definitions of Operating Modes

2.1 Batch (B)

This mode of operation is distinguished by the fact that each user-originated machine service request is fully-prepared and submitted as a single entity, and the user has no communication with the system until he receives a single report (usually printed) after completion of his computation service.

2.2 Time-Sharing (TS)

This mode of system operation is also called by several other names, including conversational, interactive, colloquy, and foreground, as contrasted with "batch".

Its distinguishing characteristic is that computer usage preparation, requests, and returns of results occur as a continuing chain of events with elements of each interspersed among the others without any predetermined sequence.

2.3 Quick-Batch (QB)

We will consider here also a special kind of batch operation in which short-time-demands users are favored by the system executive in sequence of access to system resources. Requests for machine usage of one second or so of central processor time usually are serviced and completed within a few minutes. (In traditional large-scale batch-type systems that serve all users in arrival-time sequence, response time may be hours or days.)

The QB mode of operation, which has been implemented for some hardware systems, has the advantage that response of the system to user request is quick enough so that his train of thought is often not seriously interrupted because of waiting time, as it would be if delay were many hours. It thus has, to a lesser extent, one of the outstanding advantages of time-sharing operation.

At the same time, QB has the major advantages of conventional batch operation in that:

(a) the user is encouraged to prepare each request meticulously and without artificial time-pressure, since he is not tying up terminal equipment while preparing a request for service.
(b) the user need not make himself available at the exact time when "his turn" comes up for usage of a remote terminal (keyboard) unit; he may deposit his prepared request for service at any time that may be convenient for him.
(c) the system is probably simpler and cheaper than time-sharing.

3. Student Support Considerations

3.1 Response Time

In some student circumstances it will be advantageous for the system to respond to a request within a few minutes. This is certainly the case when the user wishes to get a quick check on the adequacy of program corrections of minor nature, such as source language syntax corrections.

In other cases students might leave a request while en route to class, and be unable to use results until several hours later.

In all cases, there does not exist a directly-visible economic consideration that is always present in large-scale professional-user-support facilities: in the professional environment, system response delay often results in productivity-reduction for entire organizations, which can amount to hugh implied costs for even small delays.

In sum, there will be significant value in a batch system response time that is in the QB or few-minute category. There is not, however, strong justification of an economic nature for response speed typical of the time-sharing systems (i.e., on the order of one second for trivial service requests).

3.2 Interactivity

Modern time-sharing systems permit a high order of continuing communication between a user and a program being created or tested. This interactivity greatly speeds up user progress at two extremes of level of debugging complexity:

(a) For trivial-level (clerical or grammatical) error correction, and
(b) For advanced experimental programming, as in cases where the user actually makes experimental changes in program method or problem-solving approach.

For intermediate levels of program testing, such as in insertion of trial values of numbers, user time savings are less marked as compared with batch use of a system.

The general question of whether or not interactive system capability is of sufficient value in student learning to justify significant added costs (complexity, greater system overhead, etc.) is of great current interest. Surprisingly, then, it does not seem to have been given much attention in the research community. Approximately two dozen journal papers and research reports have appeared. Unfortuately, no paper of which we are aware includes results of a large-population controlled experiment. In particular, we are aware of no definitive experiment performed for measuring the relative pedagogical value of batch and time-sharing operation.

Those papers that address themselves directly to interactive-vs.-batch operation are somewhat guarded in their assertions as to generality and applicability of their results. Where quantative results have been indicated for learning rates, machine time consumption, and other economic aspects, most studies have shown advantages in favor of time-sharing that have been modest (i.e., typically less that 2:1).

We conclude, thus, that interactivity has not been demonstrated to have, for the support of student populations, the dramatic advantages that it can provide in some kinds of professional environments.

3.3 System Support Capabilities

(Remark: We will not discuss in this section aspects of system mechanization, such as multiprogramming, which are intended to improve system internal efficiency. We will confine this discussion to aspects of system support features that are visible to and that interface with the user.)

Many large-scale time-sharing systems have provided a somewhat more comprehensive and advances array of support capabilities than have typical batch-processing systems.

For example, powerful text-editing capabilities, usable for (perhaps only for) source-language programs are usually provided to all time-sharing users. Such capabilities are necessary if some of the primary advantages of system interactivity with the user are to be realized. In batch systems, simpler, at least partly manual, program-change techniques are adequate and are generally used.

In the area of program testing, also, more powerful software tools are justifiable, and are usually provided, with interactive systems. This is particularly true for tools that provide direct communication between the user and his program while it is operating, such as dynamic snap generators having external parameter control during execution.

Program translators in a time-sharing system may include, in addition to the conventional compilers for general-purpose procedural languages, a number of "incremental" or "conversational" compilers. These processors are designed to provide for modification of source-language programs with a minimum of difficulty and cost, at some sacrifice in object program execution economy (typically, in both time and space); at the cost of added constraints, including that of object program size; and perhaps with some loss of generality and/or language compatibility as compared with conventional processors. In the limit, a processor that performs translation (compilation) of each statement at execute time (i.e., classical "interpretation") permits any statement to be modified at any time during a pause in execution, provided only that the statement being modified does not contain a loop control index that is currently unequal in value to its starting value.

Finally, utility program support from the executive system in an interactive facility must be more comprehensive than must that of a batch system. Considerable automation on input/output control facilities, including flexible format generation controlled by inference of data space requirements, and adaptability to various kinds of I/O devices and data representations, are required for effective interactive operation.

None of the requirements outlined above is related to the nature of the user (except, of course, that in a professional environment lack of needed facilities would result in greater visibility of economic losses than would be the case in a student environment). Consequently, these requirements may in principle be considered to hold unequivocally for a student support system.

To sum up, an interactive system, if it is to realize the advantages that interactivity can provide, must include, in addition to all of the system support capabilities in a good batch system, substantial additional capability.

3.4 Program Input Preparation: On-Line Key-In vs. Off-Line Card Preparation

We consider that one major disadvantage for most users of an interactive system that primarily uses typewriter-type terminals is the requirement of program orgination by on-line key-in.

This process usually makes inefficient and low-value use of a potentially highvalue device, namely, a two-way I/O terminal. Futhermore, the keyboarding is often performed by a person whose clerical-task performance is low.

By comparison, off-line preparation of punched or marked cards (or some other input medium) may be performed without tie-up of costly and powerful on-line devices, and consequently permits more flexibility to the user as to time and procedures for this process.

The primary advantage of on-line input is, of course, the ease and flexibility with which changes or small insertions can be made to programs, data, or control information already in the system. This becomes particularly evident with use of unusually terse languages such as Iverson's APL , and/or for advanced experimental use of a system in which facility for substantial user intervention in machine processing has been developed.

3.5 Student Program Storage: On-Line vs. External

Because a primary advantage of interactive systems is ease of program change, accompanied by ease of program loading/unloading by the user, a feature that is universally supplied in such systems is at least short-term on-line program storage, without facile capability for saving of programs on-line in inactive status, together with easy and quick recall, use of the system would be inconvenient and inefficient.

This feature is valuable in any kind of facility, but in the case of a batch system, on-line user program storage is merely a convenience rather than a requirement. Let us view it, then, from a strictly economic point of view.

In a student environment, on-line user program storage may be economically attractive if suitable hardware (mass secondary memory) is already included in the system and if the effect of number and size of student programs does not require drastic cost increase for program storage.

Economic aspects of this matter are discussed in appendix O. It is concluded that on-line storage of all student programs is economically infeasible, but that in selected cases it would be feasible and is recommended. A procedure is suggested for teacher-controlled access to on-line storage of selected student programs.

If on-line storage of student programs is not provided in a batch system, each program must be read into the system from an external medium (such as cards) each time it is tested. This input requirement has been taken into account in the proposed system, and in the case of all but a few large schools the required number of terminal stations per school is one. In most cases the decision as to whether or not student programs must be read into the system at every system entry will not change the number of stations required. Thus, input/output hardware cost is not substantially affected by this requirement.

4. Economy

4.1 Central System

Costs for central system operation should be expected to be highest for time-sharing system, intermediate for quick batch systems, and lowest for conventional batch systems.

The basic rationale for the highest-cost expectation for the time-sharing system is that it requires more complex and costly hardware and software. Also, many more system control interactions must occur than batch systems; consequently, for a given level of language and function capability, system overhead costs should be expected to be larger in a time-sharing system.

The quick-batch type system (which perforce has multiprogramming capability) can, we feel, be only marginally greater in cost and complexity than the conventional batch system, but should be considerably more efficient in operating; this is because the multiprogramming capability provides automatic means for utilizing hardware elements during times when, in a conventional batch system, such elements would be idle.

Early multiprogrammed systems experienced substantial system overhead time losses, to the extent that their operation was often less efficient than would be the case for single-task operation. Advances in software and hardware technology have now resulted in solutions to this problem, such that typical present-day multiprogrammed systems are substantially more economical in operation, for a broad range of kinds of workload, than single-task systems of comparable capacity.

4.2 Terminal Hardware

Typical choices for types of terminal hardware will result in many more devices being required for a time-sharing system than for a batch system (either conventional or quick-batch) with unit cost being higher, but total cost lower, for the batch terminal hardware. For the GLC recommendations, for instance, the total of TS terminal and communications costs for the suggested system was $1.97 million per year while the total of remote batch terminal and communications cost was $1.42 million per year.**

Recent developments in medium-speed, low-cost terminal hardware* have increased the cost advantage of batch hardware, provided certain compromises in convenience and printing quality can be accepted. We feel that this class of hardware is completely adequate for typical student usage, and that the cost advantage is large enough to require that serious consideration be given to use of this type of equipment for student-support systems.

4.3 Communications

Large-population remote service over leased wire facilities will be lower in cost for either kind of batch system than for time-sharing systems, assuming heavy usage of full-time communications circuits in each case.

This is because many more stations are required in the time-shared case, while cost per communication circuit for these lower-speed terminals is less than proportionately cheaper.

For long-distance service, TS systems may lessen the cost difference by use of added equipment to provide for multiplexing of many devices on a single higher-speed line.

The basic cost advantage of the batch systems appears to remain, even with use of such equipment for TS systems.

---

* See Appendix E.

** Unfortunately, GLC did not present separately costs for the communications services.

4.4 Consumable Supplies

The cost of consumable supplies appears to be lower for a time-sharing system than for either kind of batch system.

Punched or marked card blank stock represents a substantial cost for the proposed large-population-of-users batch-processing system. (We estimated $1,300,000 per year). Cards would not be required at all for most usage of time-shared systems by student users.

Printer paper for student programs and other output should be significant in cost for any of the system types named. We estimate $50,000 per year for quick-batch. We feel that paper cost would be lower in a time sharing system.

Our reasoning is that interactive operation, efficiently utilized, makes repeated use of already-printed output and requires most accesses to print only incremental information. Batch operation, on the other hand, requires comprehensive output information to be printed for each system usage.

If there is significant service delay, advanced users are apt to request several alternative kinds of information at each entry. We feel that this slightly increases the extent to which batch exceeds TS in paper cost in a student environment.

We recommend that students purchase their own card stock; this is discussed in Appendix N.

4.5 Manpower

We feel that the central system manpower requirements will be essentially the same for all three systems but that remote terminal manpower requirements will differ considerably for these systems.

In the time-sharing environment, there would be a large number of remote consoles in constant use. This, we feel would require that groups of terminals be monitored by trained personnel. We estimate that at least 200 skilled people would be needed to monitor and assist users of 1000 consoles.

The proposed quick-batch processing system would only require enough manpower at the remote terminals to perform the simple tasks of deck and printout handling.

We recommend that manning of the quick-batch remote terminals be by voluntary student personnel. Many secondary schools have arranged for students to be excused from study periods to work as laboratory assistants, messengers, etc. Using similar procedures, selected students could be used as remote terminal equipment operators. Paid operators would be costly: For a school year, with two individuals per I/O station, paid the minimum wage of $1.60 per hour, the yearly remote station labor cost is $260,000 per year.

APPENDIX B -- Comparison of Multiprogramming Batch Systems as Proposed by GLC and IBM

1. Introduction

Comparison of Multiprogramming Batch System as proposed by GLC and IBM must of necessity be primarily a presentation of GLC's efforts.

GLC has proposed that a Multiprogramming Batch System be adopted, and specified such a system; IBM stated that such a system would be economically infeasible and did not report on detailed consideration of such a system.

2. Quotation from IBM Report

A brief discussion on a Multiprogramming Batch System was presented by IBM in their report. The following is quoted directly from IBM's report, Appendix A, pages 2 and 3:

> The conversational system used for the comparison was configured with a single medium-speed processor using 200 remote terminals in the schools. Of these, 50 contained card read/punch and keyboard/hard copy capabilities; 150 contained only the keyboard/hard copy capability.
>
> The remote batch system was configured with two-speed processors to perform the workload and meet the same throughput requirements. For this system, 100 remote terminals were configured with card read/punch and keyboard/hard copy capabilities. In addition, 50 keypunches were included.
>
> The annual total cost of data processing equipment and communication facilities for the remote batch entry system is 89% greater than those for the conversational system, when supporting the workload projected for the USOE model region.
>
> Annual cost (Data Processing and Communications Equipment only) for USOE model region:
>
> Conversational (TS): $1,176,918/year
> Remote Batch : $2,230,144/year

3. Discussion of GLC MP Batch System

GLC's batch workload estimate was similar to their time-sharing workload estimate. Program size for the "simple programming languages" is assumed to average 1400 characters; programs written in an "advanced language", 2800 characters. Advanced language programs would be written primarily by students in the higher grades, and would usually be larger in size. The number of runs required for programs in the advanced language was estimated to average 6, as compared to 3 for those programmed in the simple language. Taking into consideration the differences in the programming languages and the number of runs required for each, GLC predicted that the total number of runs the system would have to handle in a school year would be 2,819,000.

GLC recommended one card reader capable of reading 100 cards per minute and one line printer capable of 300 lines per minute for each school, except for the 4000-population schools which would require two reader-printer pairs. In addition, keypunching equipment was recommended at the rate of one keypunch per 400 students.

4. GLC Configuration Recommendation

GLC's central computer system equipment is presented in the following table:

Central Processor

Memory size (characters)	530,000

Secondary Memory

Disk Units

Number required	3-4
Capacity per unit (millions of characters)	150-200
Access time (milliseconds avg.)	150-250
Transfer rate (characters/second)	100,000 to 200,000

Must provide total capacity of approximately 600 million characters

Drums

Number required	1
Capacity (millions of characters)	2
Access time (milliseconds avg.)	15-25
Transfer rate (characters/second)	at least 500,000

Magnetic Tapes

Number required	4
Transfer rate (characters/second)	50,000 to 100,000
Density (bits/inch)	200,556,800

Card readers

Number required	1
Speed (cards/minute)	800 to 1200

Line printers

Number required	1
Speed (lines/minute)	800 to 1200

Card punches

Number required	1
Speed (cards/minute)	200 to 300

APPENDIX C--Comparison of Time-Sharing Systems as Proposed by GLC and IBM

1. Objectives

Examination of the IBM and GLC proposals for time-shared data processing systems designed to provide computational support to the same school population discloses a discrepancy of substantial magnitude in the minimum configurations deemed necessary by the respective authors.

One major difference between the two reports is in the recommended numbers of remote terminals. IBM recommends 214 terminals (156 terminals for student use and 58 terminals for administrative use); GLC proposes 978 terminals, of which none have been specifically identified as administrative-use-only. Explanation and evaluation of this large difference requires reference to the original assumptions, analyses, conclusions, and recommendations.

This Appendix provides a side-by-side presentation and analysis of the proposed systems, and attempts to reconcile the apparent gross differences.

The comparative analysis is presented as discussions of Workload Assumptions, Hardware, Software, and Estimated Costs. In addition we offer a Summary containing our overall conclusions and a concise tabular presentation of pertinent data on workload and equipment.

2. Comparison of Workload Assumptions

Both IBM and GLC, in calculating anticipated workload, used the 100,000 student population stipulated by USOE. The actual workload represented by these hypothetical students (and administrators) however, was apparently interpreted differently by IBM and GLC. Some of the factors taken into account in the reports in deriving workload estimates (and, consequently, console and processing requirements) are the following:

a. Student workload (number of console sessions).
b. Console time per problem solving session.
c. Program size.
d. Input/Output data rates.
e. Overload provisions.

These factors were estimated as follows:

2.1 Student Workload

Workloads were expressed in different units: IBM estimated 674,400 problem solving sessions per year plus 104,400 data processing sessions, or a total of 778,800 console sessions per year.

GLC estimated approximately 660,000 problems assigned for the school year for all students in the 100,000 population, with each student working alone; resulting requirement is 2,058,000 console sessions per year.

This 3:1 difference in console sessions, GLC:IBM, directly affects the number of consoles needed.

IBM removed data processing from the daytime console workload, proposing that these problems be run in overnight batch mode. IBM proposed to reduce the number of console sessions per year by providing that students be paired 80% of the time and be allowed to work individually only 20% of the time.

In order to compare IBM's proposed system with GLC's proposed system, we must multiply IBM's console sessions per year by a factor of 1.8 to account for GLC's not proposing pairing of students. (Remark: We do not here address ourselves to the question of relative pedagogical effectiveness of pairing vs. separate operation of consoles, but merely note that difference in usage recommendation contributes directly to the disparity of conclusions on facilities required.)

In IBM's Problem Solving Workload Table, the figures are projections of use after five years. We have assumed that, having neglected the 9th grade in the tabulation, IBM is distributing the 100,000 student population evenly over grades 10-12 in order to retain a 100,00 base population of active computer users. If we assume 32,000 students per grade for grades 10-12 and a total population of 4,000 students in grades 13-14, a check on the projected number of problem solving sessions per year may be obtained. This amounts to 1,061,800 sessions per year, which is an increase of 1.57:1 over the 674,400 previously estimated by IBM. This scales the number of terminals required. From context, it appears that the lower figure was used in determining the required number of terminals.

2.2 Console Time per Problem Solving Session

Both IBM and GLC state that the average time for the problem solving sessions will be "twenty minutes". Allowance for those students who will consume time beyond the scheduled limit and for the time it takes the next student to seat himself at the console and get organized will result in less than 100% utilization of the student consoles and will therefore increase the number of consoles that are required.

Inasmuch as IBM has not given sufficient data to determine how the number of terminals has been calculated, but the results are consistent with 20 minutes gross time. We have assumed that they used a twenty minute figure without "allowances" in their calculations.

GLC's calculation for the number of terminals required, while not fully detailed, shows that they have included allowances for lost terminal time. We infer from arithmetic on GLC's specifications for the Time-Sharing System that they use an average of 34.2 minutes total time per problem solving session in determining the number of console terminals required.

This difference in lost time assumptions affects the required number of consoles by a factor of 1.71:1, GLC:IBM.

2.3 Program Size

IBM's estimate of average program size is 25 statements per program while GLC's is 75 statements; thus, IBM's lower estimated student console requirement has been based on a program unit that is one third that of GLC's.

2.4 Input/Output Data Rates

Input/Output data rates based on terminal equipment selected are slightly different and therefore have some effect on data transmission and system operation. IBM's selection of Type 2741 and 1050 terminals with data transmission rates of 15 characters per second enhances their transmission capabilities over GLC's 10 characters per second with teletype ASR 35 terminals, but will not, however, improve transmission capabilities by the apparent 50%. Computer to student console transmission will be 15 characters per second for IBM's 1050, but transmission from student console to computer is limited by individual student keyboard abilities and cannot be expected to reach even the 10 character per second limitation of the ASR 35. Thus, the faster IBM terminals affect total data rate by some what less that 1.5:1.

2.5 Overload Provisions

Another factor that deserves attention is the overload capabilities of these systems. IBM has removed administrative data processing from the student consoles and has stated that these programs are to be batch processed in the evening. Administrative terminals selected by IBM have card handling abilities, but these are slow in transmission rates and are not suitable for high volume. These terminals are expected to handle not only the routine administrative functions but also batch processed student programs at the end of the day, as well as the overflow from the student consoles.

IBM has stated that if the load at remote sites is too great, the jobs could be taken by courier to the central facility where they would be processed throughout the night with results being transmitted back to the local terminal for immediate use in the morning. Several problems are thus created.

In order to have the output ready for immediate morning usage, the remote printers will be operating unattended during the night. If any station runs out of paper there will be either lost output or, at best, untransmitted output not available for use for a couple of hours. If the assumption is made that the remote printers have an extremely high Mean Time Between Failures and that a new roll of paper is placed on the printer before it is left unattended, someone must arrive at the school sufficiently early in the morning to separate all the output received during the night.

If data were transmitted continuously for a 16-hour period at the steady rate of 15 characters per second, 864,000 characters would be received. With IBM's estimate of 25 cards per program and under the assumption that each card contains an average of 30 columns, an equivalent of 1152 program listings could be received at a single terminal.

Courier service loading would also follow the peak loading of the computer; anything not completed during the day would be expected to be ready in the morning. Attempts to bypass courier service to get data to the computer for processing would force at least one administrative employee into overtime which would ultimately affect school costs.

If batch processing is to be used for overflow processing, students cannot use the special programming language that IBM has proposed. This will require students to learn an additional language in order to get work accomplished during peak loading periods. Not only must a new language be learned but also the student must be provided with a means to prepare input for batch processing. (An example of difficulties is the fact that programs would not exist in card form.)

GLC does not explicitly provide for overload; instead they make an allowance for lost time during the school day to account for various losses: 1/3 for grades 9-12 and 1/5 for grades 13-16. The comparative workload factors are tabulated in Workload Summary below.

Overall ratio of estimated console hours/year is about 5:1, GLC:IBM.

3. Hardware

Central processors have been selected for the respective systems based on the amount of processing that will be required. IBM projects a relatively light workload, and recommends a central processor of medium speed; GLC's recommendation of a much higher speed processor is supported by a larger workload estimate.

Neither of the reports have given detailed specifications on the central processor nor on the peripheral equipments, but they have given approximate speed ranges and tentative equipment requirements. The respective recommendations are summarized in the Hardware Summary table below.

4. Software

A major consideration is the choice of the problem solving language to be used for time-sharing operation. GLC proposes that students program in conventional languages such as FORTRAN, COBOL, and ALGOL. IBM suggests, instead, that a special problem solving language processor be made available for student use, although there is not generally available at the present time a problem solving language with all the capabilities described in the IBM report.

IBM's calculations of console time per session and number of consoles required have been based on availability of such a language, which they (and we) hope one day will be generally available. Since it is a requirement of the present study that readily available or shortly anticipated software be used because of the firm schedule for operational status of the system, we feel that projections of number of terminals should not be based on the special language.

5. Costs

There is significant difference between the IBM and GLC reports in the matter of estimated operating costs for the system; the time-sharing system recommended by IBM would cost $1.46 million per year while GLC's would cost $2.96 million per year.

This difference is largely attributable to the number of remote terminals: IBM recommends 214 consoles (156 for student use, 58 for administrative use); GLC, 978 consoles (all for student use.)

IBM selected Type 2741 and Type 1050 terminals; GLC, Teletype ASR 35 terminals.

If IBM's system cost were adjusted to provide the number of terminals recommended by GLC, their system cost would increase by $1.41 million per year if multiplexing equipments were used or $2.40 million without multiplexing.

6. Overall Reconciliation of Console Recommendations

Let us now summarize the differences between GLC and IBM estimates as they affect the number of time-sharing consoles required to serve the student population. The factors below show how number of consoles should be modified in order to account for differences in assumptions. In each case the factor shown is the number by which the IBM estimate of console requirements would have to be multiplied in order to reconcile IBM's treatment of that variable with GLC's.

Item	Factor
Number of console sessions per student per year	3.0
Pairing of student (IBM) vs. students using alone (GLC)	1.8
Student distribution in grades 10-12 (To reconcile IBM's projected usage against console sessions, to rectify what seems to be an internal inconsistency in the IBM report)	1.57
Console utilization (IBM's arithmetic assumed 100% usage of console time. GLC's assumed that unused console time, changeover time, and other contributions to inefficiency would require 34.2 minutes total time assignable in order to deliver 20 minutes net usable time.	1.71
Average program size	3.0

If all of the above factors were truly independent, and therefore multiplicative, one would have to multiply the number of student consoles recommended by IBM (viz., 156) by 43.5 in order to reconcile the IBM and GLC recommendations, which would result in over 6000 consoles required. There is unquestionably some interaction between the factors, so that this multiplication is inappropriate.

The above summary does, however, help to display the softness of planning parameters for the two time-sharing system plans. COMPLAN feels that this should be interpreted not as a weakness of either contractor's report but rather as further evidence (supporting that in the published literature) that the present state of the art of planning time-sharing computing facilities for student support is inadequate to permit reasonably quantified system design for such a facility at this time.

7. COMPARISON TABULATIONS

7.1 WORKLOAD SUMMARY

GLC		IBM	
Grade	**Student Populations**	**Grades**	**Student Populations**
9	23,700	9 - 12	100,000
10	22,005	13 - 14	4,000
11	18,295		
12	16,000		
13	10,600		
14	8,000		
15	800		
16	600		

Grade	Assigned pbs per student/yr
9	3.96
10	4.25
11	5.11
12	6.63
13	13.5
14	13.5
15	13.5
16	13.5
Average	6.6

Student Usage

Grade	Sessions per problem	terminal hours per session	terminal hours per student year
9	4.	.55	8.71
10	3.5	.6	8.93
11	3.	.65	9.97
12	2.5	.7	11.60
13-16	3.	.5	19.7
average 9-16	3.12	.57	11.8

Terminal hours
per 1.8 student-years,
average: 2.5

Time per console session		----------------------
Useful time	20 minutes	20 minutes
Total time	34.2 minutes	
Total problem workload	660,000 problems/year	----------------------
Total console usage	2,058,000 console sessions/year (1 student/session)	674,000 console sessions/year (average 1.8 students/session)
	1,180,000 console hours/year	224,700 console hours/year

7.2 HARDWARE SUMMARY

	IBM	GLC
Central Processor	Medium Speed	High Speed
Memory Size (characters)	512,000	1,000,000
High Speed Secondary Storage		
Device	Disks	Drums
Storage requirements (millions of characters)	230	6
Access time (milliseconds avg.)	100	15-25
Transfer rate (characters/second)	125,000	500,000
Low Speed Storage		
Device	Data Cell	Disks
Storage requirements (millions of characters)	400	652 (includes 517 for admin.)
Access time (milliseconds avg.)	600	150-250
Transfer rate (characters/sec.)	40,000	100,000 - 200,000
Magnetic Tapes		
Number required	2	4
Transfer rate (characters/sec.)	30,000 - 180,000	50,000 - 100,000
Density (bits/inch/track)	200 - 1600	200,556 and 800 (1600 desirable)
Card Readers		
Number required	1	2
Speed (cards/min.)	1000	800 - 1200
Printers		
Number required	2	3
Speed (lines/min.)	1100	800 - 1200
Card Punches		
Number required	1	1
Speed (cards/min.)	300	200 - 300
Optical Mark Readers	4	---

APPENDIX D -- Multiprogramming System-Analytical Discussion

1. Introduction

a. Definition of terms

Multiprogramming is the time-sharing of processors (s) by a number of not necessarily related programs simultaneously present in main memory; the number of programs may be larger than the number of processors.

Multiprocessing is the use of two or more self-sufficient processor units with a single logically continuous and jointly addressable memory.

A macro is a (hardware or software) instruction, execution of which has the effect of executing several simpler instructions.

Input/output tanking refers to the time and storage requirements for queued I/O buffer manipulation, where available buffer pools are placed in a "tank" and the tank can be drained and refilled as required by an I/O supervisor.

b. Priority Classes

A. Administrative Query
B. Student-Waiting
C. Student
D. Student-Deferred
E. Administrative

2. Factors in Scheduling from Queue

The general mode of operation of the system will be to read programs from the card readers into memory and to schedule compilations from the queue in memory. The header card of each program will be required to contain certain information about the program including its priority class, estimated running time, and output volume.

The executive system should have the capability to scan the jobs waiting in each priority-class queue and to execute them in some order other than the sequence in which they were entered into that queue.

For jobs of equal priority, it should assimilate the header card information, run short jobs before long ones, and those with a low output volume before those with a large amount of output. If all other things are equal, then the jobs should be run in the sequence in which they are in the queue.

Student jobs will usually be short in estimated running time and have low estimated output volume; students should get reasonable turnaround time of only a few minutes after the early morning peak has been processed.

Another factor should be considered in the scheduling algorithm: Should the system be capable of multiprogramming compilations, executions, and input/output processing? It is recommended that the executive have the capability to do multiprogramming, but that the operator be able to exercise control over the extent to which multiprogramming occurs. Input/output multiprogramming must occur if the remote users are to receive responsive service. However, large scale multiprogramming of compilations and executions may result in excessive system overhead and lead to less effective use of system resources. For this reason it is recommended that the system have an option to process priority B and C jobs in a high speeed sequential batch mode thereby reducing system overhead for such things as moving program segments or pages to mass memory.

Other factors which sometimes may be of importance in job sequencing include the number of tape units needed for the job and available at that time, the amount of mass memory available for input/output tanking, and the operational factor of which form is on which line printer at that time.

It may be advisable to try out several sequencing schemes before arriving at management policies for the cooperative facility. At different times of the semester it may also be necessary to change the sequencing scheme to meet peak requirements for educational or administrative tasks. Therefore, it is desirable that the sequencing algorithm provided by the manufacturer be simple, fast, and easily modified.

3. Background on Utilization of System Resources

Generalized multiprogramming capability has come to be recognized as a potentially valuable tool for increasing system operating efficiency. In most second generation systems, even the most highly refined types of programs have seldom utilized as much as 70% of central processor time; that is, the central processor is actually computing less than 70% of its observed productive time. Careful observations on large working systems have disclosed effective utilization of central processor time, averaged over the computing day, between 12% and 50%; hence, an improvement by a factor of 2 or more may be available through more efficient utilization techniques without increasing processor internal speed.

However, as noted above in Section 2, multiprogramming can result in excessive overhead. The promise of higher effective utilization of the central processor by multiprogramming in a number of current time-sharing systems has not materialized because of the large amount of time spent swapping programs in and out of core memory and waiting for execution of disk/drum orders.

In the proposed system, hardware/software planning must provide for adequate data traffic capability.

4. Graceful Degradation

Multiple processors may lead to improved system efficiency. There are other advantages in providing redundancy of all resources of the system. With redundancy and proper system design, it would be possible in most cases to avoid having the system go out of service in the event of failure of any unit, even a central processor.

With 50 or more users tied to the system, a failure means that 50 or more jobs will have to be identified as having been involved in the system failure and will have to be processed again. However, there is clearly a trade-off between system cost as a result of providing the redundancy and service to the users.

In an environment where many jobs run for long periods of time and/or where system collapse would have unacceptable cost and/or danger consequences, system redundancy would be a requirement. In the school environment being considered, we feel that full redundancy is not necessary.

There will be a certain amount of redundancy provided by the essential system resources, such as the number of input/output channels, and the number of high speed printers at the central facility.

To meet the system workload requirements, it is possible that some vendors will propose multiprocessor configurations. The system executive should take advantage of such redundancy as does exist: it should be possible for the operator in charge to withdraw individual redundant units without causing the system to go out of service.

Because of the higher cost, it is not recommended that the entire system be provided with from the start. However, as the workload begins to grow and some experience is gained on the impact of system failures on the effective use of the system, the need for redundancy should be reevaluated. The system design should permit addition of redundant resources without the need for reprogramming on the part of the users. That is, the system should be capable of modular growth as well as modular withdrawal of resources.

A related consideration, which becomes significant in the case of a system fault (such as the loss of an area of memory containing active-status-list information) that cripples the executive system itself, is that of system rollback and restart. In order to guarantee restart capability after an unscheduled and perhaps disastrous interruption of operation, it will be desirable to perform periodically a total dump of main memory, critical tables in fast secondary memory, and all processor registers.

It is also necessary to enforce system conventions that avoid the possibility of irrecoverable faults; thus, for example, a previous-state copy of a file being updated must be preserved, together with source data for the updating, until a later reservemaster file has been confirmed as correct and has been placed in protected storage.

Frequent system dumps to conventional types of tape or disc equipment were, in the past, prohibitively expensive because of the excessive time required; the present availability of high-data-rate auxiliary memory devices permits frequent dumps at reasonable cost.

5. Communications from Remote locations to Central Facility

There will be times when it will be necessary for an operator at one of the remote locations to communicate with the operator at the central computer facility. While at times this may necessitate a phone call, at many other times a short message sent over the already existing line to the central facility by means of a card (s) placed in the card reader that will print out on the operator console at the central facility should suffice. Likewise, the operator at the central facility should be able to send a message to a selected remote location or to all remote locations onto the line printer (s). The executive system should have this capability built into its basic design. Some means of calling attention to such messages, such as preceding them with a highly-visible printed pattern, would be desirable so the messages will be noticed promptly and any required action can be taken.

6. System Planning for Multiprogramming and Multiprocessing

The purposes of multiprogramming and multiprocessing in the context of our system are to provide sufficient, but not excess, system resources to accommodate input from 57 or more remote card readers and output to 57 or more remote printers, and at the same time to keep the central processor (s) occupied with useful work as nearly continuously as possible. Thoughtful users realize that a typical computer job includes substantial central processor wait time. If input/output is not well buffered, as has too often been the case in some older computer systems, lost main frame time may be so great as to be visually noticeable at a diagnostic console; such a circumstance cannot be tolerated in this system.

It is within the state-of-the-art today to accomplish all input/output in a multiprogramming mode and reduce effect of that particular reason for inefficient use of the central processor. To do this will require adequate hardware to provide sufficient input/output tanking space, which is an essential additional system resource if the large number of remote stations are to be handled satisfactorily.

There are several conflicting factors which affect the memory size that is optimal for a given system in its workload environment:

a. Techniques for increasing the effectiveness of main memory utilization in the system (to be discussed later in this Appendix)

b. The growing power of mass memory facilities; their effective use instead of tape for system service functions not to be kept in main memory; and the fact that multiprogramming permits reloading of service routines to be overlapped effectively with other work, now permit some functions that have typically been permanently retained in core to be brought in as required from mass memory

c. The increasingly complex nature of executive systems and the many additional buffer spaces needed for input/output in all-devices-on-line systems are now driving upward the amount of memory space needed for executive functions, despite factors (a) and (b) above.

d. The increasing use of mathematical and data processing techniques that use large memory space very effectively

e. Multiprogramming systems, in attemping to improve processor utilization, need concurrent access to many jobs in main memory

f. The cost of high-performance memory is decreasing relative to the cost of processors and other parts of large systems.

Factors (a) and (b) tend to reduce main memory space demands while all of the other factors tend to make larger main memories desirable and practicable. The net result of these several factors has been a substantial increase in the typical size of memory available for large multiprogramming systems, whether with single or multiprocessors. By allocating a fixed area of memory for student programs and exercising some control of multiprogramming of other jobs, it is believed that the size of main memory can be held to 1/8 million words or 1/2 million characters.

We next consider the general approach of assignment of system resources. In a multiprocessor system, a generalization of the single processor system, any number of fully independent processors up to the system design limit can operate within a single, jointly-addressable memory. Each processor may assume the executive function, and must do so in order to obtain system service, to write in memory areas reserved for executive purposes, or to obtain reserved information. Thus, each processor must be able to operate in at least two somewhat different modes, which may be called "executive" mode and "job" mode. Note, however, that the important executive function of "sequencing", or determining what computation or job segment is to be performed next, must be performed by only one processor at a time. There are other hardware requirements that are desirable for efficient operation of a multiprogramming/multiprocessing system:

a. Direct addressing of large memory (by character in the case of character-string-manipulation instructions).

b. Processor status information stored and automatically transferred by hardware when change occurs in order that interrupt response time be short.

c. Positive memory protect for each job with respect to its currently assigned memory area.

d. Dynamic relocation of programs and data without significant use, for this purpose, of processor time.

e. Capability for controlled access, to reentrant programs and to (read only) Common Routines (one copy of each) and commonly-accessible data, from any number of calling programs.

f. Ability to use conveniently a hierarchy of main memories.

Other system resources, in addition to the central processors, should also be assignable by the executive system in an anonymous manner. The general concept is to have each system resource which lends itself to this manner of operation be assigned to a "pool". When the executive system recognizes that there is a need for the given resource, it looks at its "pool" of that particular resource to see if one is available. If one is, it assigns it to the job, marks it as being in use, and goes back to the next task for the executive. If none are available, it places that request in a queue, goes on with the next executive task, and returns a short time later to see if by then one of the needed resource is available. There must be enough of each of the system resources placed in such pools to keep the queues short and then wait for them also short or the system will bog down in long queues waiting for the particular resource that is in short supply. However, this concept of operation permits some economies in the amount of system hardware that is needed, and by virtue of the flexibility of assigning resources in the pool concept, the system is seen to be highly modular in design and lends itself well to some realization of "graceful degradation" advantages without unacceptable added cost.

The system should be able to use the ASCII*, and possibly also include as standard a set of 4-bit (binary coded decimal) character manipulation commands, primaily to meet the needs for administrative programs. If a 4-bit subset is implemented, it should be the ASCII 4-bit one. Since the system will involve extensive use of data communications, careful consideration of the commercially available services of the communications agencies is a necessity from the start, paying particular attention to compatibility of equipment at interface points with regard to character sets and codes.

In a multiprocessor facility, in order to perform processor assignment, the executive system must keep current in its own active lists information on each processor's status at the time of the most recent execution of the assignment algorithm. In order to permit selective interruption of processors in consideration of their current status between points of communication with the executive system, however, it isnecessary that a processor currently in "executive" mode be able to query, and if necessary, interrupt all other processors. Therefore, if a multiprocessor system is proposed, inter-processor communications commands must be available in the system.

Reference has been made previously to "reentrant routines", (also called Pure Procedures, Common Procedures, and Single-Copy Routines). Reentrancy provides that subroutines,

---

* American Standard Code for Information Interchange (USASI - X3.1-1966)

to arbitrary nesting depth, and segments of large routines such as compilers and executive routines that are called upon to perform service for many users, can be accessible to any number of calling routines in memory. Execution of a reentrant routine may be interrupted before normal exit, and the calling program may later request proper continuation of the calculation even though both calling program and program may have moved meanwhile. The provision of this capability is desirable in a multiprogramming environment with a large number of users of compilers.

The provision of adequate buffering and input/output queue control has not been discussed previously and must be properly provided for if the service to users is to be acceptable.

Assuming that both the card readers and the printers at the remote locations are all being driven at about 200 cards per minute and 250 lines per minute respectively, there is a need for 450 accesses per minute per terminal pair. For the 57 terminal locations we therfore have a requirement to perform 450x57=25650 accesses per minute or about 430 accesses per second. Access times for commercially available fast drums are such that access rates of that magnitude will require drum-order-resequencing in order to permit several accesses per revolution. Hence, the drum access rate may be a limiting factor for efficient queue handling. A possible solution to this problem would be to use some form of bulk core memory.

Let us now consider I/O tanking space requirements. If we store only one incoming and one outgoing program for each station with peak (end-of-semester) output of about 300 lines per program and 30 characters per average line, we have, ignoring added space for control information:

300x30=9000 characters per program

for output tanking requirements. Thus for storing one output job for each of the 57 remote locations, there is a requirement for about .5 million characters of storage. Input tanking of perhaps 5,000 characters for one job at each station brings this total space requiremnet to about 0.75 million characters.

This would not provide for any queueing of input or output. We feel that the system should provide for short I/O queues of 2 or 3 for each remote location, which increases the fast secondary memory requirement to at least 2 million characters. Since, however, the anticipated workload on the system will not be as great as the worst-case numbers used in the above arithmetic, we feel that an initial complement of 1 million characters of fast secondary memory, provided the system permits future expansion, will suffice for reasonably efficient operation.

It is possible that bulk core memory will be available, adequate in its performance predictability, and even competitive in overall cost with high-performance drum hardware that has an advanced hardware/software feature for drum-order-resequencing to increase access rates.

We are concerned, however, that the number of manufacturers presently offering bulk core memory is limited and that firm specification of such hardware would result in limiting the number of vendors who would be able to participate in a competitive selection.

Therefore, although we are including in this report indications of the quantity and type of hardware that we estimate is needed we recommend that, in the request for proposal to be submitted to vendors, only the operating performance parameters be specified in the case of fast secondary memory.

In order to make efficient use of the printer at the remote locations, several operational mode options for return of output must be available.

For jobs with a large volume of output or those requiring wide-paper format, such as some administrative jobs, it must be possible to designate that output is to go on the high speed printer at the central facility.

For jobs which will be returned on the remote printers, it should be possible to provide for printout sequence control of the queue for each remote printer.

It is anticipated that the usual mode of operation will be to queue all output in auxiliary memory for later printing, since this will permit more efficient use of the printers for jobs which call for a significant amount of calculation between each line of output. An added benefit of storing output for later printout is that if the printer must be taken out of service for repairs, replacement of paper, and so on, the processing of jobs does not need to be discontinued.

Experience in many computer facilities has been that the growth of the workload seems to follow an exponential pattern. It is anticipated that the same phenomenon will be experienced in the proposed system. The hardware and software must therefore be adaptable to an orderly growth of reprogramming.

APPENDIX E -- Input/Output Station Hardware

1. INTRODUCTION --- CONCEPT: BATCH

As discussed above, the system concept chosen as economically preferred is that of multiprogrammed, priority-with-brevity batch operation.

In this kind of system usage programs with data are presented as fully-prepared "runs" recorded in a machine-readable medium, and results are presented to users as monolithic output reports.

2. INPUT

2.1 Input Media

Among the input media that might be used are conventional Hollerith-format punched tabulating cards. Many other machine-readable media were considered in the present study, ranging from direct student usage of on-line keyboard (i.e., no "medium" at all) to magnetic tape produced by keyboard devices.

In consideration of system and user economics and flexibility, the tabulating card has been chosen as the preferred input medium.

2.2 Preliminary Choice: Punched/Marked Cards

The actual medium to be used for the proposed system will be tabulating cards that are printed with timing marks across the bottom edge so that they can be either punched or pencil-marked in Hollerith-coded representation of alphanumeric characters for photo-optical reading by the combination-type readers described below.

The cards may, because the readers are synchronized to the timing marks and because an asynchronous communications interface (Bell System 202D) has been chosen, be punched or marked in either 80-column (i.e., conventional) or 40-column format, or even in a mixture of the two.

The 40-column format uses the locations on the card of alternate columns of the 80-column format; i.e., if one numbers columns according to location on the card, corresponding numbers would be as follows:

Column	Numbers
40-column	80-column
1	2
2	4
3	6
4	8
5	10
6	12
etc.	etc.

Cards may be punched conveniently in "40-column" format merely by setting a conventional keypunch to skip every odd-numbered column of the 80-column format.

This is pointless, however, in view of the ability of the readers to read the 80-column cards. The purpose of the 40-column format is to provide for reasonable use of human usage when pencil-marking data in Hollerith format: this spacing seems not to require unreasonable care in card-marking, while the 80-column spacing is quite difficult to mark without using extreme care.

The determining factor in how the reader interprets the card is the placing of the timing marks. Thus, if a card is to be partially prepunched or otherwise keypunched, in a specified field, this may be done at 80-column density if the timing marks are appropriately placed.

For student usage, in order to avoid the complication of a multiplicity of card formats, and in view of the fact that only a few cards in a typical FORTRAN program exceed 40 columns in length, we recommend that the "standard student card" be preprinted with a full-length 40-column-density timing mark comb. (For the occasional card that goes beyond 40 columns, of course, FORTRAN language provides for continuation on one or more successive cards.) See Appendix H for a physical sample of such a card.

2.3 Card Reader Types

The simplest and least costly card readers handle punched-cards only, in fixed 80-column density using standard Hollerith layout, and read either optically or by electrical spring contacts through the card holes.

The high-speed card readers that have been supplied for a number of years as parts of computing systems are much higher in price than the above-described devices. For the purposes of the system described in the present report, the advantages offered by these machines are not commensurate with the high prices. In particular, since more modestly-priced machines having lower speed are adequately fast for this system, in all but a few of the largest schools, it is inappropriate to pay a premium for unneeded capacity.

The chosen type of optically-sensing punched-or-marked-card reader is presently produced by two U.S. manufacturers. Their respective models are somewhat different in design and in detailed characteristics, but are highly similar in principal. Both offer operating speeds slightly above 200 cards per minute.

One of the readers offers a high-speed eject feature, actuated by sensing of a special mark that may be placed on the card immediately after the last meaningful information. This feature should speed up reading of FORTRAN program decks, which tend to average 30 or less columns per card.

3. OUTPUT

3.1 Media

Of the many media available, the only ones that were considered seriously for this project were untreated paper and electroconducting paper, to be used with ink printers and stylus-conduction printers respectively. The latter, at a cost of about 4¢ per square foot, turned out to be unacceptably expensive in the huge quantities needed for the proposed system application. Although low-priced conductive printers are available, the overall cost of system usage will be much lower with higher-priced ink printers that use cheaper paper.

3.2 Initial Choice: Medium-Speed Alphanumeric Printer

For the proposed system, we suggest the use of low priced, medium-speed alphanumeric printer. At present the ink-stream type appears to be most economical.

3.3 Printer Types

The ink printers having appropriate speed and cost ranges fall into a few groups:

(a) Impact-type (full line parallel, several-character-parallel, and single-character) printers in which the type face moves at high speed parallel to the surface of the paper, and in which momentary impact of a hammer or hammers drives the paper, an inked ribbon, and the type face together at the right instant to cause the chosen type image to be imprinted on the paper;

(b) Type-bar type, in which the type moves perpendicularly to the paper, again causing a ribbon to be pressed against the paper by the chosen type face; and

(c) Ink-stream type, in which a stream of electrically charged ink droplets is deflected by a variable-intensity and direction electrostatic field in such a way as to trace out the desired character.

Impact printers are now available at speeds up to 40 lines per second and down to about 40 characters per second. Prices range up to $60,000 for the mechanical printer assemblies without electronics.

Type-bar printers are available at speeds of from 10 to 60 characters per second; the higher-speed units are substantially higher in price than are the typewriter-speed devices, ranging up to several thousand dollars per unit. Ink-stream printers, now unfortunately available from only one maker, are intermediate in price and offer speeds up to 120 characters per second, and appear to be capable of being developed to operate at significantly higher speeds.

3.4 Ink-Stream Printer

The recommended device has been commercially "available" for several years, but has been delivered to commercial customers only recently. It appears to produce print of high contrast but low character-shape accuracy, in present production. The resulting printed copy is irregular in appearance but is highly readable and is felt to be completly adequate for the present systems's purposes. (cf. physical printout sample, Apdx. M.)

The ink-stream printer uses a total of 40 ink projector-deflector devices to print 80-column copy, character-serial. It requires no buffering electronics and has no apparent "flyback" time. The lack of need for buffering removes a costly hardware element from the printer. The no-flyback-time feature makes the printer efficient for short-line, variable-format printing such as constitutes a large part of the printing volume for typical student programs.

The fact that this device is quite simple is quoted by its maker in support of a claim for high reliability and low maintenance cost. His commercial maintenance prices are quite low; although they have not yet been quoted publicly, these prices appear to confirm the high-reliability claim.

APPENDIX F -- Nature of Billing Problems

The IBM and GLC studies both were hampered by the lack of readily available "hard data" on student-support experience. The system used for logging information for billing purposes can also be used to provide other aspects of "hard data" for a detailed analysis of the characteristics of the workload. The log should keep records of the following items of system resources:

1. Processor time for compilation
2. Elapsed clock time for compilation
3. Processor time for execution
4. Clock time for start of input
5. Total elapsed time between start of input and end of output
6. Mass memory time and space used for each job.
7. Amount of core or fast access memory required by the program (this may be in characters, word, or "pages" depending upon the system and how it functions.)
8. Number of input records (probably cards) for a batch mode operation
9. Card reader start and stop time for each student job
10. Number of lines of output
11. Job identification of the entry in the log.

The job identification for users should be set up in such a way that it is possible to associate a particular run to an individual student, and that re-runs on a given problem by an individual student can be identified. From such "hard data" the number of runs per problem per student and the amount of system resources used by students can be analyzed.

This will also provide a vehicle to control excess usage of the system by some students. While it is desirable to encourage students to use the computer on problems of their own interest and choosing, experience has shown that control must be exercised to avoid excessive usage and misuse.

This is accomplished in part by requiring on the header (initial) card of the program deck provision for limits on processor time and number of lines of output. Such a technique does not control such things as excessive number of debugging runs or too many self-generated projects without faculty supervision.

The collection of the recommended data will provide a basis for control if a periodic report is sent back to the faculty, probably on an exception basis, e.g., report all students in course 123 who required more than 5 runs for Exercise 3.

The collection of all of the data recommended will add a significant amount of system overhead for the pilot system, but the data will be of great value in evaluating system performance and in planning for other similar systems for other localities. It will also permit the various schools and colleges cooperating in the use of the system to estimate utilization and costs for preparation of budgets and to receive monthly reports on the expenditure of resources to see that they are staying within the funds budgeted.

In order to make it attractive for potential contractors to bid on the operation of the central facility, it will probably be necessary to guarantee to the contractor a minimum payment per month for a specified amount of services and system time. This might be, for example, rental of the system for a single prime shift plus some overhead expenses for operators, system programmers, management and a reasonable profit. The contractor might then be allowed to sell time on a "service bureau" basis for third and possibly also a second shift.

For the first few years it is anticipated that penetration into the schools will be low, but will gradually increase. It may be that for the first year even the prime shift will have system resources available that could be sold and a portion of the money returned to the schools.

It is beyond the scope of the present report to go into greater detail on such possibilities, but they will need to be spelled out in the request for proposals to vendors to operate the system.

The following billing methods are proposed for batch-mode and time-sharing mode systems:

1. Batch-Mode

It is assumed that the system logging routine has recorded the items specified above. A monthly price for the proposed system service can be determined from computer and other direct and indirect costs to the operating contractor.

For the basic billing for a school to participate in the project it is proposed that the student enrollment at the school be the basis for determining the school's pro-rata share of the basic monthly system cost. The number used for student enrollment could be the number used as the basis for federal aid from the preceding year.

Thus for a school with an enrollment of 800 students in a total school population of 100,000 students, the school's pro-rata share would be 800/100,000 of the system cost. This would buy for the school an entitlement to 800/100,000 of the system resources.

If that school used time in excess of its pro-rata share of the system resources, it would be billed at the rate set by the system operator for excess usage. This could occur, for example, if penetration of computer usage in one school builds up faster than at some of the others.

In computing charges for mass memory, both time in storage and the amount of storage space used are the basic factors to be used. For a character-oriented system, the unit of measurement for this type of storage might then be called the "character-day". One school might use up its share of this system resource by storing a large number of characters for a few days, while another might store a lesser number of characters for the whole month. This type of storage would be relatively unimportant for the student usage since in a batch mode system students would not normally be storing programs or data. The computation of the monthly billing for a particular school would be done using a basic formula related to the individual system resources as follows:

Define:

T_{cpu}	Central processor time used (hours)
T_{mm}	Time mass memory is used (hours)
MM	Amount of mass memory used (may be in characters, words, or pages depending upon the system)
IO	Input/output channel time (hours)
a	Unit cost of central processor time
b	Unit cost of mass memory (e.g., word hour)
c	Unit cost of input/output channel time

$$Cost_{school}=(T_{cpu})(a)+(T_{mm})(MM)(b)+(I/O)(c)$$

This would be compared to the school's share of 800/100,000 of the monthly system cost. If it exceeded the pro-rata share, the school would be billed for the excess. If it were less, the school would still have to pay the pro-rata minimum share, unless the operator had been able to sell time to commercial users to reduce the overall monthly cost of the system for that month.

If the system permits recording of usage of main memory at little cost, one might consider another term in the cost formula to account for the amount of memory used by the program. Otherwise, the charges for main memory should be taken into account in establishing the monthly rate for central processor time. For student usage this term would probably not be meaningful since student programs would usually be of small size. For administrative applications the term could be an important one. A compromise billing technique would be to make no charge if the program used less than 1/50 of the amount of main memory, and to bill on a sliding scale for additional memory in increments of 1/50.

A further discussion of economic aspects of student program storage in on-line mass memory is given in Appendix O.

The various costs of controlling student access on the basis of checking cost of each current student job against the past record of that student and his current "bank balance" was considered. We feel that the overhead costs associated with the file space required, the search time to find and check the student records, and to update them after each run, and the increased complexity of the job accounting function of the executive system are prohibitive.

It is recommended that the student usage accounting program should be run on a daily basis. It should update a file containing all individual student usage data. Students who exceed their allowable costs should be identified and reported on an exception basis. Input to this updating process should provide for approved "grants" of increased allowable costs for individual students at faculty discretion.

2. Time-Sharing Mode

In addition to the usage items mentioned at the start of this appendix (excluding, of course, batch mode entries such as card reader time for input from the remote locations), in the time-sharing mode the amount of time a particular console has been logged into the system should also be recorded.

In the commercial world, time-sharing service bureaus generally charge for console time as well as for the use of other system resources. Such a billing technique is not recommended for the school system.

Instead of using the number of students in a school to establish its pro-rata share of the system costs, as recommended for a batch system, the number of time-sharing consoles in the school as compared to the total number of consoles in the entire system should be the basis for computing the school's monthly minimum billing.

The reason for not using console "on" time for a billing item is that the consoles will be a critical resource at most schools and will be logged "on" for almost all of the allowable prime shift time.

If such is the case, it is unnecessary to cost-account for console time, because the number of consoles in the school divided by the total number of consoles in the system would result in the same pro-rata share for that school.

Information on users at the consoles should, however, be recorded. A record of console time, user identification, and central processor time used, can be analyzed to identify inefficient users on the basis of an unusually low ratio of processor time to console "on" time, excessive amount of console time to solve a particular problem, etc.

In a time-sharing mode, the system will have to have a large amount of mass memory available for storage of programs by students. The parameter "character-days" introduced in the batch mode discussion would again be the basic unit for charges for programs stored in the mass memory. The billing formula for the time-sharing would be quite similar to the batch mode, and might be as follows:

T_{cpu}	Central processor time used (hours)
T_{mm}	Time mass memory used (days)
MM	Amount of mass memory used
FM_i	Amount of fast memory used (words) during the time period i

CPU_i	Central processor time used during time period i
T_{co}	Time console "on" (hours)
a	Unit cost of central processor time
b	Unit cost of mass memory
c	Unit cost of fast memory
d	Unit cost of console time (if used*)

$$Cost_{School} = (T_{CPU})(a)+(T_{mm})(MM)(b) + c\sum_{i=1}^{n} F M_i CPU_i + T_{co}(d)$$

The individual schools should each pay their own costs for hardware, staff and operating supplies at the individual installations. See Appendix G for a discussion of communication costs. In order to plan for similar systems at other locations, the participating schools should report their local costs to some central collection point for consolidation and analysis.

An examination of billing techniques in use for time-sharing installations reveals no generally accepted procedure. The discussion which follows presents two examples of techniques currently in use.

2.1 The RAND Corporation JOSS System

The following discussion is based on the discussion in "JOSS: Accounting and Performance Measurement" by G. E. Bryan, Memorandum RM-5217-PR, June 1967.

Charges are made for use of the JOSS system on the basis of only three system resources: (a) the amount of compute time used, (b) the amount of core required for program and data, and (c) the amount of file space used on the disc and the time it is retained.

In addition to recording the amount of cpu time used and the amount of core required, the billing system of JOSS has two basic requirements: One, to create a record when an item is discarded from the disc file, and the second, at the end of a calendar month to write charge records for all files on the disc at midnight on the last day of the month. The following information is needed in these records:

1. Disc-Discard Type
   (a) File name or other identification
   (b) Size of file
   (c) User identification
   (d) Date file write, date last used, and date discarded
   (e) Identification of user who discarded the file

2.2 End-of-Month Type
   (a) File name or other identification
   (b) Size of file
   (c) User identification
   (d) Date file written, date last used, and date of end-of-month

In the proposed school enviornment the "date last used" entry will be useful for improving the discipline of users of the system rather than for billing purposes. The amount of mass memory available can be expected to be a critical system resource, yet it will be difficult to get users to remove information that has outlived its usefulness. In the student environment, this could be expected to be a problem.

The human tendency seems to be to "save it. I may need it some day." On the mass memory there will not be room for such a luxury and it will probably be necessary to periodically purge from the mass memory all files that have not been used for some period of time. Thus, although the entry "date last used" may not be used on a day-to-day basis for billing, it will be used periodically to purge idle files and in that respect will serve to reduce billing costs for non-productive files.

As a second example of an approach to billing on a time-sharing system, we considered the method being used at the University of Michigan. It is still in a developmental stage and will undoubtedly undergo more changes before evolving to a more or less stable form. A formula is being used, the principal aim of which is to charge the user for only that part of the computer which he actually uses while he is using it. The items in the formula include the time that a terminal is connected to the computer system, central processor time used, time that file storage is used, pages of file storage used, pages of virtual memory used (changes dynamically with time), number of cards read, number of cards punched, and number of lines printed (not on a terminal). Using the foregoing items and the unit cost of each, charges can be computed. One formula serves for both batch and terminal processing, but there is one basic difference in the accumulation of time. For terminal processing the timing factor takes into account the entire connect (i.e., telephone) time, whereas for batch processing the comparable term accrues charges only when the program has control of the central processor (i.e., CPU time).

For the same problem assumed solvable in either batch or terminal mode, Michigan's experience indicates it will cost more to solve it in terminal mode than in batch mode, perhaps by factor of at least 2 or 3 times as much. The ratio of costs is heavily dependent on the type of problem being solved, the software system being used, etc. This is in agreement with COMPLAN's judgment that a system designed to operate in the time-sharing mode will be more costly than a batch mode system.

---

* Although the last term is not recommended, some charge relating the imput and output demanded by the user is appropriate, and a term for the input/output channel usage as in the batch mode formula might be substituted.

Appendix G -- Communication Economics

1. Introduction

Communications requirements are considerably different for time-sharing and for multiprogrammed batch operation.

In the time-sharing environment there are many remote terminals in the system. Each terminal may be serviced by a low grade line or by a combination of low grade lines, multiplexing equipment, and high grade lines. System configurations will vary depending on the number of terminals originating at a single location requiring termination at another single location. The number of terminals needed, equipment selection, system configuration and communication distance affect overall costs. Communication equipment proposed by GLC for 978 terminals would cost $323,000 per year for their assumed average line length of 30 airline miles.

For typical metropolitan region reaching out to a 100 mile radius, we feel that the average line length from the central facility will be about 12 miles. For this distance, with half the lines extending across a state boundary, communications costs for the batch processing system as recommended by COMPLAN would be $150,000 per year. (Remark: Had we estimated 30 miles average, as did GLC, the communications costs for the recommended system would increase to $235,000 per year.) In addition, there is a one-time installation charge of $12,000.

Unit costs for communication service items are tabulated below.

2. Time-sharing Communication Costs (per GLC report)*:

System element	No. units	Unit cost $ per month	System cost $ per month	
Multiplexor	57	295	16,800	
Data-Phone terminals	114	44	5,000	
Communications line to central facility av. 30 miles @ $3/mile (per GLC**)	57	90	5,100	
Total Communication Costs:				$26,900 per month $323,000 per year
Communication Cost per Student Year				$3.23

---

* GLC did not separately tabulate terminal hardware and communications service costs, but we have separated these in order to clarify comparative system economies.

** Local rates vary considerably, especially between interstate and intrastate service. COMPLAN feels that GLC's rate estimates should have been increased to an average of approximately $117 per month for interstate service and $130 per month intrastate. As noted below, we feel that the 30-mile average line length estimated by GLC is high for a typical metropolitan region. In view of these effects which tend to compensate each other, and considering that we have not recommended nor analyzed a time-sharing system, we present the final GLC data verbatim for comparison with the recommended multiprogrammed-batch system.

3. Multiprogrammed Batch:

For the following calculations it is assumed that 29 installations are intrastate and 28 are interstate.

System Element	No. Units	Unit cost	System cost
One-time installation charges:			
202D data set (4 per reader-printer station)	228	50.00	11,400
Service points (4 per reader-printer station) intrastate (no charge interstate)	116	7.00	800
Total One-Time Installation Costs:			$12,200
Monthly communications service costs:			
202D data set (4 per reader-printer station) intrastate (29 stations)	116	40.00	4,600
interstate (28 stations)	112	30.00	3,400
Communications lines (C1-conditioned), 12 miles average length, 2 lines per reader-printer station: intrastate (29 stations)	58	53.50	3,100
interstate (28 stations)	56	40.00	2,300
Total Communication Service Cost:			$13,400 per month
		or	$161,000 per year

If we assume an average line length of 30 miles, as GLC did, the Communication Service Cost estimates would become:

202D data sets - intrastate	4,600
202D data sets - interstate	3,400
communication line - intrastate	5,900
communication lines - interstate	5,700
30-mile average Total Com. Svc. Cost:	$19,600 per month
or	$236,000 per year

APPENDIX H -- Financial Support Considerations

1. Introduction

It is COMPLAN's understanding that the U.S. Office of Education's intent in supporting the present study is to create a system design and supporting software (computing and educational) that will permit future replications of the pilot system to be economically viable without continuing Government support.

To this end, the Government intends to plan and demonstrate a workable and economically attractive system, which for productive use should then be capable of being supported by State and local educational agencies.

2. Government-Support Recommendations

We recommend that Federal Government support be provided for the following aspects of system creation and related activities:

2.1 Overall planning and development of the hardware/software system, including related curriculum planning and support materials;

2.2 System and user performance research activities, to yield observations that will be useful in future usage and improvement of this and similar systems for educational purposes;

2.3 Teacher training, at least to the point of existence of an initial cadre of teachers who have been introduced to computing concepts and their potential applicability in the educational community;

2.4 Some sort of Government support for voluntary cooperative teacher activities among teachers of the same or similar subjects, with a view toward future existence of libraries of teacher-created demonstration (computer) programs;

2.5 Planning and procurement of broadly useful follow-on developments, especially of software and application systems that may be useful throughout the educational community.

3. State and Local Support Recommendations

We feel that the following aspects of future (i.e., replications of the pilot system) installations should be supported by other than Federal funding:

3.1 System operations, including the cost of computing equipment as reflected in the cost of machine time, for student-usage eductional activities as well as for teacher and administrative usage;

3.2 Installation, including building modifications and incidental costs such as furniture and storage facilities for materials;

3.3 Maintenance of facilities and computing hardware, including the remote input/output stations;

3.4 Consumable supplies, including all supplies used at the Data Central facility (but not including card stock to be used by students, which we recommend be a student-supplied material just as pencils, notebooks, etc. are in general student-supplied);

3.5 Communications services for data transmission and related telecommunications such as incidental telephone (voice) service required by system usage.

4. Operation of Central Facility by Commercial Firm

Consideration has been given to the way in which the Data Central facility should be financed, managed, and operated in order best to serve the educational community at minimum cost.

Many kinds of organizations could operate such a facility. Perhaps the first kind of organization that comes to mind, in view of administrative and other complications associated in dealing with different school districts that are likely to be in different states as well as in different counties, is the non-profit special-purpose laboratory controlled by an independent management board. While such an organization clearly could perform the operations function, we feel that it would be less economical and responsive than a profit-oriented organization.

A basic consideration is that the Data Central facility recommended in the present report is capable of operating and producing valuable results on at least a 20-hour-a-day basis, while the educational-support process requires that most of its demands be satisfied during the normal school day.

Consequently, the computing facility would have many hours of time in the 168-hour calendar week in which its services would usually not be required.

We feel that disposition of the unused computing capacity is essentially a commercial type function that can be served best and most economically by a profit-oriented organization.

It is evident that a contractual arrangement would be necessary under which individual school districts would be guaranteed a specified level of support for their educational and administrative requirements, on some budgetable pricing basis which would include minimum dollar commitments to the operating contractor. The agreement would have to include means for adjusting billings upward to account for increases in usage over anticipated limits, and downward to permit the school districts to gain some financial advantage from commercial success by the operating contractor in profitably disposing of excess computing capacity.

Appendix I -- Operations

1. Introduction

Operations includes the handling of jobs as they are submitted for processing, the placing of the jobs in a desired sequence for input to the card reader (usually called scheduling the workload), the loading of the program and data decks in the card reader and their later removal, the set-up of tapes, disc packs, etc. on the system peripherals, the handling of output from the system, and finally the return of the finished computer output to the user.

2. Data Central Operations

We consider first the operations function at the Data Central facility. Operating guidelines must be defined and available to the operations staff. This is usually done in the form of a written operations manual. While it is expected that a contractor will operate the central facility and his staff will be responsible for performing the operations function, many of the procedures and policies that will be contained in the manual will be the result of policies and philosophy of the participating schools, so we will go into some detail on the contents of such a manual.

Input to the central facility for direct processing will normally be via courier service. Courier routes and schedules must be established; vehicles must be acquired and personnel hired to drive them, including provision for emergency conditions such as vehicle breakdown or employee sickness.

Procedures for log-in and scheduling of jobs are needed. In the multiprogramming environment some job mixes will be found to run more efficiently together than others. Therefore, the senior operator or shift supervisor must be given some authority to adjust the job schedule to take advantage of opportunities to improve processing efficiency.

A central tape/disc library will be needed to handle tape and disc holdings and the standard source program punched card library. There must be established procedures and policies for making back-up copies of data files, how long to retain the older files, how many generations of a file to keep, records of activity of files and periodic review to remove obsolete files, tape cleaning and maintenance, and establishment of routine procedures for handling jobs which require tapes or disc from the time the job is submitted, through the scheduler to the librarian, the computer operator and finally back to the user, with provision for return of old and new tapes/discs to the library. Carefully thought-out procedures and well-trained staff can save setup time on the system and insure smooth work flow in the computer center. A job run form must be designed for submission with the job to assist the operations staff in setting up and processing the job. Color coding of the form to identify jobs which need tapes, disc packs, special output forms, etc., can simplify the handling process.

An effort should be made to achieve some form of standardization among the participating schools in design of multi-school special printer forms for similar jobs. Certain administrative tasks will be common to all schools; by pre-planning, it is reasonable to expect that a single form could meet the needs of all. If such can be accomplished, minor savings can be achieved by central procurement of forms and by reduction of setup time for the printer. More important savings could result from common usage of a few well-planned programs.

The alternative would be a multiplicity of forms, of formats, and of programs and file structures. Each school would probably have to keep its own forms and send them to the central facility for each run.

(Remark: If all or many schools use common forms the production control people handling the ouput must be careful to assure that the output is routed properly to its rightful recipient, particularly in the case of limited-access reports such as salary administration.)

The central facility should have necessary equipment to handle any bursting and binding of printer output that is required.

Physical design of the central facility must include adequate space to perform the functions noted above, and must provide room for expansion.

3. Remote (In-School) Station Operations

Operating procedures and policies for the remote stations must be carefully planned to assist the school faculty, administrative staff, and volunteer student workers. In contrast to the operations staff of the central facility which will be made up of people whose primary occupational specialty is data processing, the operations staff at the remote locations will be teachers, clerical workers, administrators and students, most of whom will be inexperienced in data processing.

A training program will be needed for student volunteers to serve as operators and to handle the production control function.

An operations manual will be described below for use by the operators and production controllers. Student workers must be thoroughly familiar with the contents of the manual before they are permitted to work independently. Proper equipment handling techniques will increase equipment life and improve system performance. Adherence to school policies and rules will improve the efficiency of the production control function and thereby give all users better service.

Particular attention must be given to the development of good judgement on the part of the students to ask for help when it is needed. Students who try to correct problems on their own without adequate knowledge or proper tools are likely to cause damage and create additional problems.

To facilitate supervision of the remote station equipment and to permit ready access to it for administrative queries, it is desirable that the remote station equipment be placed in a room in close proximity to the administrative offices.

It should be anticipated that there will be considerable student activity involved in the submission of jobs and retrieval of output, so it may not be desirable to place the equipment in the main administrative office, but rather in an adjacent area. Some building modifications may be necessary in order to provide a counter to physically separate the card reader, page printer and operator and production control workspace from the students submitting or picking up computer runs. The counter will keep unauthorized persons away from the equipment and must be sufficiently large to accommodate lines of students at peak submission and retrieval times. Adequate walkspace leading up to the counter is also needed to minimize congestion in the area.

Individual schools must decide whether or not students may wait for their programs to be run (contingent also on system loading at the time and whether turn around time of a few minutes is possible.) Adult behavior on the part of the students will induce administrators to allow students to wait. In such cases, an area nearby equipped with desks might be provided, thereby making it possible for students to study while awaiting their output.

When sufficient faculty become trained in computer programming, it might be reasonable to establish a "computer study hall" in a room adjacent to the remote equipment location where students could wait for their runs and ask for consultant assistance from the computer study hall teacher when necessary. In many colleges advanced computer science students perform such consultant services for novice programmers and the foregoing would provide an analogous service in the secondary schools. Without such a service, teachers of programming courses and other courses using the computer may be forced to spend an inordinate amount of time in class answering student questions on their programs thereby decreasing the amount of class time available for teaching new concepts.

Some form of adult supervision of the remote station equipment will be mandatory. This could be provided by the computer study hall teachers, if there is such, or by office administrative personnel. The supervisor's function will be to assist student operators as necessary and to exercise disciplinary control over students if required.

4. Remote Station Operations Manual

The operations manual mentioned above must be available at each remote station for use by the student operators, production control staff, and supervisory personnel. Content of the manual must include information and instructions on the following:

4.1 Routine Equipment Procedures

(1) Placing cards in the card reader and making it ready to read them
(2) Making the card reader "not ready"
(3) Removing cards from the card reader
(4) Making the page printer "ready" or "not ready" for printing
(5) Putting a new roll of paper in the printer
(6) Adjusting paper alignment or sprocket hole alignment in the printer
(7) Removal of paper "jams" in the printer
(8) Removal of card "jams" in the card reader

4.2 Abnormal or Emergency Procedures

(1) A list of known abnormal symptoms of the system, their probable cause, and what steps the operator at the remote site should take. If the student operator and administrative personnel cannot identify the problem and/or correct it, it will normally be advisable to have an instructor who is more familiar with the system operation check the symptom and identify possible ways to correct the trouble. If the trouble is beyond the capability of the instructor to fix, a maintenance call should be placed.
(2) The names and phone numbers of the maintenence personnel must be readily available so that a call for repair service can be placed promptly at any hour.
(3) For emergencies, information on how and where to cut off the power to the card reader and printer must be readily available, perhaps even displayed on or near the equipment, so the power can be shut off quickly in case of serious malfunction.
(4) For suspected or observed malfunction of the central facility hardware or software, name and phone number of the central facility computer room or maintenance room must be included.

4.3 Staffing Procedures

(1) List of qualified student volunteers should be posted, showing name, period, and locations index so substitutes or extra persons can be obtained in case of absence or unusual workload.

It is assumed that student volunteers will be adequately trained so they can do any job at the remote location. While the staff at the central facility would be hired by the contractor running that facility, the staffs at the remote locations would be selected and supervised by the individual schools.

One way that such technical and semi-technical student labor has been supplied in the past at the high school level has been through a Laboratory Assistant program. Student Lab Assistants would typically volunteer to work one class period each day for the instructor of a course in electronics or a laboratory science. These students generally receive little or no credit for this class period; they become Lab Assistants for the privileges and further educational opportunities that the Lab Assistant program offers.

In the CUES program, a Computer Laboratory Assistant position could be set up in cooperation with school officials; the position could be open to any interested student who would volunteer one period per day for the operation of the terminal units. The entire program could perhaps be placed under the direction of an instructor of electronics, data processing, or office practices.

The primary advantage of the Computer Laboratory Assistant plan is the considerable cost reduction realized through the use of volunteer student labor. This seems to be the only practical plan for largescale computer use at the high-school level.

(2) List of qualified instructors by name, period, and location index to provide assistance in case the student and administrative staff are unable to cope with the difficulty.

4.4 Supplies

(1) How and where to get a new roll of printer paper
(2) How and where to get blank cards for source program preparation, header and control cards, etc.
(3) How and where to get new ribbons for the keypunches

4.5 System Messages

(1) A section which identifies all standard messages that the system may output on the page printer, their meaning, and any action the operator must take in response to the messages
(2) A section which tells the operator how to initiate messages from the remote location to the operator at the central facility

4.6 Operating Policies and Procedures

(1) Priorities of various types of jobs that may be submitted and instructuions on who to contact for a decision in the event that priority of a job is not readily determinable
(2) List of duties of each person working at the remote location, who is in charge of all the student workers, and what authority he has to make decisions
(3) Advice on keeping the work moving through the system as rapidly as possible. This may include advice on balancing the workload to facilitate multiprogramming
(4) Safety rules, especially if the cover of any piece of equipment is removed to correct such things as a card jam or other minor difficulty.
(5) Sample copies and instructions on when and how to complete any forms or records that operators or other student volunteers must complete. These might include reports of unusual system behavior, a record of when a particular piece of equipment broke down, what appears to be wrong with it, what attempts were made locally to fix it, who was called from the maintenance shop to fix it and when the call was placed, when the repairman arrived, when the equipment was fixed and put back in use again. Such records may seem laborious to make, but can pay good dividends when one is trying to establish that maintenance service is not what it should be.

The operations manual will become the "bible" of the staff of the remote location. It should be easily expanded to add new sections as the need for them becomes evident.

Appendix J -- Workload

1. Introduction

System workload is of primary importance in determining such items as central processor parameters, communications facilities, throughput capabilities and system cost.

System workloads should be based on actual collected data applicable to this endeavor; however, data of this nature is not available in sufficient quantity within the realm of secondary school computer usage. It has, therefore, been necessary to use the estimates of GLC and IBM, checked against U.S. Air Force Academy observations, to provide a crude prediction of the anticipated computer usage.

The system workload will consist of student work from computer-concepts and data processing courses, student problem-solving for a variety of applications in other courses, and aministrative tasks. In addition, we anticipate a limited amount of experimental work on computer instruction techniques and use of the computer by teachers in their classrooms.

2. Anticipated Morning Queue

It is expected that student submission of problems upon arrival at school will produce an early morning workload peak. Additional small peaks will occur as students change classes and a somewhat larger peak will follow lunch hour. In addition, there will be a more or less constant input from students throughout the day as they come from study halls and possibly from laboratory sessions of programming courses.

An additional early morning peak of high priority administrative tasks (absence and tardy lists, substitute teacher queries, etc.) can be anticipated on a daily basis. The net result is that a sizable system queue can be expected every morning.

3. Scheduled Demands

Some balancing of the workload for administrative tasks will permit more effective use of the system resources and will tend to stabilize turnaround time for student work. If the workload is not balanced by scheduling some administrative work to be processed in a multiprogramming mode with student jobs throughout the day, a queue of long administrative jobs with requirements for the high speed printer, tape units, etc., may cause undesirable delays in processing this part of the workload because of queueing on various system resources.

Estimates for system requirements have taken into account the U.S. Air Force Academy program statistics. The 25-card typical program size suggested by IBM is considered by COMPLAN to be much too low. The GLC estimated average of 75 cards is considered reasonable. However, since the system must be designed to handle end-of-semester peak workload rather than "average", calculations are based on 125 cards per program. (Remark: It is our informal observation that several student-support systems now in use that can handle average student workload appear to develop disastrous queues, with turnaround time increasing to many days, during the last third of each student semester. It is our assumption that this bottleneck results from development of a workload peak and that such service delays cannot be tolerated.)

4. Student Program Listings

COMPLAN notes the possibility that there may be a need for program listings of student programs which would be made via the remote stations. This possibility was overlooked by both IBM and GLC and would add considerably to the workload on the remote station equipment and on the communications portion of the entire system.

If the need for program listings materializes, it could result in the necessity for additional system resources to handle queue buffering for input/output tasks. For example, the GLC estimate was that 660,000 problems will be assigned per year for the entire student population. If each student asks for an average of only 2 page listings per problem, this would add another 1,320,000 jobs to be processed.

5. Penetration

The extra jobs would not put much of a workload on the central processor, but would all have to be handled by the production control staff at the remote locations and would utilize almost the same amount of time on the input/output equipment as an actual run.

In view of the lack of IBM data or estimates of workload for remote batch processing, COMPLAN has chosen to use the GLC estimate of 660,000 problems per year to be solved using the computer. This number of jobs relies heavily on the accuracy of the tables in Chapter 6 of the GLC report. COMPLAN questions the realism of the penetration factor estimates and recommends that further study be given to refining the penetration factor table. The overall workload estimates should be recalculated after the new values for the penetration factors have been obtained.

6. Administrative Workload

While primary attention has been given to the student workload, the administrative load must not be overlooked. We feel that most of this workload will be handled via courier to the central facility. Our calculations assume that the administrative workload will be processed throughout the day in multiprogramming mode concurrently with the student workload.

The GLC estimate of 3-6 hours of administrative printer time per day is felt to be high for the first year or two, but that this workload component will grow steadily if the service is predictable and adequate.

Most of the demands for system resources from the administrative load will be related to use of input/output equipment, especially in connection with file processing. Consequently, the main implications of the administrative workload are that there must be enough secondary memory to handle the on-line file requirements and enough modularity in system design to permit addition of peripheral equipment and memory as the need arises.

Appendix K. SYSTEM COST MATRIX

	PURCHASE COST			LEASE/YEAR			PURCHASE/YEAR		
	(100K) Pilot System	100K Student Replica	200K Student Replica	(100K) Pilot System	100K Student Replica	200K Student Replica	(100K) Pilot System	100K Student Replica	200K Student Replica
1. HARDWARE									
1.1 Data Central									
1.1.1 Central Processor	$600K	$600K	$600K	$180K	$180K	$180K			
1.1.2 Main Storage	800	800	800	240	240	240			
1.1.3 Bulk Core or Fast Drum	500	500	600	180	180	240			
1.1.4 Mass Memory	400	400	500	180	180	200			
1.1.5 Magnetic Tape Units	250	250	300	80	80	100			
1.1.6 Other Peripherals	150	150	200	60	60	70			
1.1.7 Communications	300	300	400	100	100	120			
TOTAL DATA CENTRAL	$3.0 M	$3.0 M	$3.4 M	$1.02 M	$1.02 M	$1.15 M	$0.72 M	$0.72 M	$0.816 M
1.2 Remote Stations									
1.2.1 Card Readers	$245K	$245K	$490K	$ 59K	$ 59K	$118K	$ 59K	$ 59K	$118K
1.2.2 Printers	296	296	593	71	71	142	71	71	142
1.2.3 Manual Punches	390	390	570	94	94	137	94	94	137
TOTAL REMOTE STATIONS	$932K	$932K	$1653K	$224K	$224K	$397K	$224K	$224K	$397K
1.3 Data Communications									
1.3.1 Installation Costs	$ 12K	$ 12K	$ 24K	$ 3K	$ 3K	$ 6K	$ 3K	$ 3K	$ 6K
1.3.2 Communications				161	161	322	161	161	322
TOTAL DATA COMM.	$ 12K	$ 12K	$ 24K	$164K	$164K	$328K	$164K	$164K	$328K
1.4 Hardware Maintenance									
1.4.1 Data Central				$120K	$120K	$140K	$120K	$120K	$140K
1.4.2 Remote Stations									
Technicians				38	38	65	38	38	65
Vehicles				6	6	10	6	6	10
Spare Parts				16	16	30	16	16	30
TOTAL HARDWARE MAINT.				$180K	$180K	$245K	$180K	$180K	$245K
TOTAL HARDWARE				$1.588M	$1.588M	$2.12M	$1.288M	$1.288M	$1.786M
HARDWARE COST/STUDENT/YEAR				$15.88	$15.88	$10.60	$12.88	$12.88	$ 8.93

Appendix K. SYSTEM COST MATRIX(cont'd)

	PURCHASE COST			LEASE/YEAR			PURCHASE/YEAR		
	(100K) Pilot System	100K Student Replica	200K Student Replica	(100K) Pilot System	100K Student Replica	200K Student Replica	(100K) Pilot System	100K Student Replica	200K Student Replica
2. SOFTWARE									
2.1 Design and Implementation									
2.1.1 FORTRAN Resident Compiler	$200K								
2.1.2 Oper.Sys.&Util.Routines	300								
2.1.3 Library Programs	100								
2.1.4 Adapt Exis.FTN Compiler	50								
2.1.5 Adapt Exis.ALGOL Compiler	20								
2.1.6 Adapt Exis.COBOL Compiler	70								
2.1.7 Adapt Exis.BASIC Compiler	40								
2.1.8 Adapt Other Compilers	50								
2.1.9 Adapt Admin.Package	70								
TOTAL SOFTWARE DES.&IMPLEM.	$900K	$100K	$100K	$216K	$ 24K	$ 24K	$216K	$ 24K	$ 24K
2.2 Maintenance Direct Labor									
2.2.1 Programmer Analyst	$ 18K								
2.2.2 Senior Programmer	15								
2.2.3 (Two) Programmers	22								
2.2.4 Clerical	5								
TOTAL SOFTWARE MAINTENANCE	$ 60K	$ 50K	$ 70K	$ 60K	$ 50K	$ 70K	$ 60K	$ 50K	$ 70K
TOTAL SOFTWARE COST				$276K	$ 74K	$ 94K	$276K	$ 74K	$ 94K
SOFTWARE COST/STUDENT YEAR				$2.76	$0.74	$0.47	$2.76	$0.74	$0.47

	PURCHASE COST			LEASE/YEAR			PURCHASE/YEAR		
	(100K) Pilot System	100K Student Replica	200K Student Replica	(100K) Pilot System	100K Student Replica	200K Student Replica	(100K) Pilot System	100K Student Replica	200K Student Replica
3. SPACE									
3.1 Central Site Preparation									
3.1.1 Bldg. Modification	$ 35K								
3.1.2 Raised Flooring	20								
3.1.3 Extra Air Cond.	12								
3.1.4 Furniture	4								
3.1.5 Decorating	2								
3.1.6 Misc.	2								
TOTAL CENTRAL SITE PREP.	$ 75K	$ 75K	$ 85K	$ 18K	$ 18K	$ 20K	$ 18K	$ 18K	$ 20K
3.2 Central Site Space Rental				$ 25K	$ 25K	$ 30K	$ 25K	$ 25K	$ 30K
TOTAL SPACE COSTS				$ 43K	$ 43K	$ 50K	$ 43K	$ 43K	$ 50K
SPACE COST/STUDENT/YEAR				$0.43	$0.43	$0.25	$0.43	$0.43	$0.25
4. OPERATIONS PERSONNEL AND VEHICLES									
4.1 Data Central Personnel (Direct Labor)									
4.1.1 Manager							$ 22K	$ 22K	$ 22K
4.1.2 Shift Supervisors							18	18	18
4.1.3 Computer Operators							42	42	42
4.1.4 Tape Librarians							14	14	21
4.1.5 Prod. Ctl. Clerks							20	20	25
4.1.6 Keypunch Operators							5	5	10
4.1.7 Clerical Workers							10	10	15
4.1.8 Couriers							8	8	12
TOTAL DATA CENTRAL DIRECT LABOR							$139K	$139K	$165K
4.2 Vehicles				$ 6K	$ 6K	$ 12K	$ 6K	$ 6K	$ 10K
4.3 Remote Stations							$ 0	$ 0	$ 0
TOTAL OPNS PERSONNEL AND VEHICLES							$145K	$145K	$175K
OPNS PERSONNEL & VEHICLES/STUDENT/YEAR							$1.45	$1.45	$0.88

Appendix K. SYSTEM COST MATRIX(cont'd)

	PURCHASE COST			LEASE/YEAR			PURCHASE/YEAR		
	(100K) Pilot System	100K Student Replica	200K Student Replica	(100K) Pilot System	100K Student Replica	200K Student Replica	(100K) Pilot System	100K Student Replica	200K Student Replica
5. SUPPLIES									
5.1 Non-consumables									
5.1.1 Disc-Packs	$ 39K	$ 39K							
5.1.2 Magnetic Tape	17	17							
TOTAL NON-CONSUMABLES	$ 56K	$ 56K	$100K	$ 13K	$ 13K	$ 24K	$ 13K	$ 13K	$ 24K
5.2 Consumables									
5.2.1 Data Central									
Paper	$ 7.2K								
Cards	1.8								
Printer Ribbons	4.8								
Miscellaneous	1.2								
TOTAL DATA CENTRAL	$15 K	$ 15K	$ 30K	$ 15K	$ 15K	$ 30K	$ 15K	$ 15K	$ 30K
5.2.2 Remote Stations									
Roll Paper	$12K								
Other Paper	30								
Cards	10								
Miscellaneous	2								
TOTAL REMOTE	$ 54K	$ 54K	$108K	$ 54K	$ 54K	$108K	$ 54K	$ 54K	$108K
TOTAL CONSUMABLES	$ 69K	$ 69K	$138K	$ 69K	$ 69K	$138K	$ 69K	$ 69K	$138K
TOTAL SUPPLIES COST	$125K	$125K	$238K	$ 82K	$ 82K	$162K	$ 82K	$ 82K	$162K
SUPPLIES COST/STUDENT/YEAR				$0.82	$0.82	$0.81	$0.82	$0.82	$0.81
GRAND TOTAL				$2.134M	$1.932M	$2.601M	$1.834M	$1.632M	$2.217M
GRAND TOTAL/STUDENT/YEAR				$21.34	$19.32	$13.01	$18.34	$16.32	$11.09

6. Discussions of Recommended System Costs

These estimates differ from GLC's Multiprogrammed-Batch estimate in three significant respects:

(a) Hardware costs for the central facility are higher because of our estimate of requirements for handling of the input/output queue buffering.

(b) While the GLC report noted that there would be additional costs for such things as supplies, building modifications, personnel, etc., their report provided estimates only for hardware (including communications). COMPLAN's estimates also cover software, space, operations, and supplies. It is evident that the costs for such items are significant.

(c) GLC proposed use of (relatively high-priced, high-performance) second-generation and reader and line printer hardware. Because of cost of the newer types of equipment (cf. Appendix E) proposed by COMPLAN is much lower, our anticipated system cost is correspondingly lower.

As computer science courses and applications penetrate the curriculum there will clearly be implications on the size of the faculty. This may vary greatly from school to school. Since this aspect is not a part of the computer system and associated peripheral requirements, no attempt has been made in this report to estimate the cost of changes in faculty. It is felt that such a cost item should more properly be related to costs of curriculum modifications.

School administrations may also be faced with a requirement to provide salary incentives in order to attract and hold teachers who are well-qualified to teach computing, in view of the heavy non-academic demand for such people.

APPENDIX L -- File Security Considerations

1. Introduction

The administrative files to be stored in the system present a new problem in that, at least in principle,* they can be accessed through any remote terminal of the system.

This kind of file security problem for administrative records, although it is well-known in large industrial and commercial installations of computing facilities, is a new problem for school administrations.

The primary file security problem is that of controlling write-access, although some data must be read-access-controlled as well.

2. Types of Access Control

Several alternatives for file security are possible. Among the more widely-used methods are:

a. Time-lock control
b. Password control
c. Use of a form of storage, such as a disc pack, which places the file on the system only when needed for a specific administrative job. This may be thought of as a flexible type of time-lock control and can be accompanied by a password system.

3. Time-Lock

Fixed-schedule time-lock control can be simple and positive but may be impracticable. It is felt that occasions will arise when use of a given administrative file will be mandatory and will not be within the time specified by the time-control feature. If the system were to provide some method for circumvention of the time-lock, then one has in effect reduced control to a form of pass-word system. A possible alternative would be to permit time-lock changes to be entered only from the central facility and to require the capability for making authorized time-lock changes.

4. Password

A password control system, while admittedly not as secure as a time-lock, would permit access to the files at any time without disruption of the processing of work. It is anticipated that access to the card reader at each remote location will be controlled by the individual schools. By using a distinctive series of job identification numbers for Administrative jobs, a distinctive color header card to make them identifiable, and allowing only certain persons to enter them into the card reader, one can reasonably control the input process. By requiring that the executive system include the capability to change the passwords, by limiting the number of people who have access to passwords, and by punching passwords on uninterpreted cards, it is felt that a password control system can provide comparable security to that present in the current situation with school administrative records stored in file cabinets in school offices.

5. Physical Removal Control

The use of removal disc packs instead of permanently-on-line files would provide a form of time-control and would permit password control as well. This approach has the disadvantage that a given file will usually not be on line; thus, response time for administrative queries would be degraded to the extent that there would be set-up time for the operator at the central facility to obtain and mount a disc pack. Also, the possibility would exist that all the units for mounting disc packs would be in use so a query might be subject to indeterminable delay.

6. Recommendations

Of the three alternatives mentioned, our recommendation is password control. Removable-file control would be second choice and time-lock, third. Our cost estimates provide for partially-removable and partially-permanent on-line files.

---

* There will be some restrictions related to format and the short (80-column) line length of the remote printers.

Physical Samples of Printer Output and Marked/Punced Cards

APPENDIX N -- Preparation and Handling of Decks and Programs

1. Preparation of Decks

1.1 Keypunching by Students

(1) The workload profile is likely to be a series of sharp peaks before school, during lunch hour, and after school, coupled with a steady base level throughout. Some peaking will also be experienced between classes, although not every student will be able to go by the computer center between each class. A student unskilled in typing could be expected to punch an average of about 1 card per minute, not taking errors in account at this time. For the grade unit it is assumed that a student program will average 100 cards. It would therefore take a student about 100 minutes to punch an average program. Using the workload estimate of Appendix J and Chapter 6 of the GLC report, the following calculations give an estimate of the number of key-punches needed to permit the students to do their own keypunching of the decks:

Grade	Student Population	Avg. Problems per student per grade unit	Equivalent Population	Cards per Program	Total Cards	Total time in minutes to punch total cards
9	300	3.96	1188	100	118,800	118,800
10	275	4.25	1169	100	116,900	116,900
11	225	5.11	1150	100	115,000	115,000
12	200	6.63	1326	100	132,600	132,600
Totals	1000		4833		483,300	483,300

Assuming the keypunches are available 8 hours per day and that the school year consists of 180 days, the requirement for keypunches is:

$\frac{483,300}{8x180x60}$ = 5.6 or approx. 6 keypunches/1000 students

The foregoing is for initial keypunching alone. We next take into account error corrections. We will use the student averages of Table 6.15 of the GLC report. However, it is necessary to recalculate the entry for runs/student to take into account the assumption that no corrections are made on the first run. Thus for 9th graders for simple language, to calculate the table entry we multiply: 2 x .9 x 3.96 = 7.1 in lieu of the 10.7 in Table 6.15 which was calculated using 3 runs. In making corrections we assume that a student is slower than in the inital keypunching, since a greater percentage of his time is spent to get seated at the keypunch, get his old run out, find the card in the deck to be corrected, etc. Calculations for corrections are then as follows:

Re-runs per problem: simple language - 2 advanced language - 5

	Grade			
	9	10	11	12
No. of students	300	275	225	200
Proportion in simple language	.9	.8	.6	.5
Proportion in advanced language	.1	.2	.4	.5
Problems per student	3.96	4.25	5.11	6.63
Runs per student in simple language	7.1	6.8	6.1	6.63
Avg. no. cards to correct per run in simple language	3	3	3	3
Avg. no. cards to correct per student in simple language	21.3	19.4	18.3	19.9
Runs per student in advanced language	2.0	4.25	10.22	16.6
Avg. no. cards to correct per run in advanced language	7	7	7	7
Ave. no. cards to correct per student in advanced language	14	29.8	71.5	116.2
Total no. cards per student to correct (average)	35.3	49.2	89.8	136.1
Total time to make corrections at 1.5 minutes per card	16,000	20,000	30,000	41,000

Estimated time needed for correction converted to the keypunch requirements is then:

$\frac{107,000}{8x180x60}$ = 1.2 keypunches/1000 students

The foregoing assumes that all machines are used 100% of the time which is not realistic because of the peaked nature of the workload and because some machines will break down and need repairs. In a student environment the latter is usually noticeably higher than when all keypunches are operated by a paid staff. Assuming the machine use factor of 2/3 from the GLC Table 6.5, our recalculation is:

$\frac{483,000 + 107,000}{2/3x8x180x60}$ = 10 keypunches/1000 students

There is one more factor we have not taken into account. The computations above for average number of cards to be corrected per run are for programming errors and do not take into account keypunch errors students are certain to make if they punch their own cards. Students are likely to have to punch several cards to produce a good card. Assuming that it takes on the average 1.5 cards punched to produce 1 good card for the initial punching of the decks and 2 cards to produce 1 good card in the correction process, the keypunch requirements finally take the form:

$\frac{1.5x483,000+2x107,000}{2/3x8x180x60}$ = 16.3 keypunches/1000 students

1.2 Keypunching by Paid Staff

An alternative approach to individual keypunching by the students is to have a paid staff do the initial punching of the student decks and to require the students to make their own corrections. In this situation, assuming the keypunch operators punch 5000 strokes per hour and the average program statement requires 30 strokes (i.e., 30 strokes per card) the data for grades 9-12 is as follows:

Grade	Cards	Strokes
9	118,800	3,564,000
10	116,900	3,507,000
11	115,000	3,450,000
12	132,600	3,978,000
	Total	14,499,000

That represents about 2900 hours of keypunch time. Thus for the school year this is a requirement for $\frac{2900}{8x180}$ which is about 2 keypunch operators/ 1000 students.

However, in dealing with averages and the total load for the year we have ignored the problem of peaks in the workload.

A possible coupling of the need for keypunch operators to a high school vocational program may be feasible in some of the schools and might solve some of the peaking problem. One might consider offering a course in keypunching as a regular course in the vocational curriculum. It would probably be too expensive to rent keypunches for an entire class, but the students could take typing as a first step, and then be given limited access to keypunches later in the course or in an advanced course for which the prerequisite was to have achived a minimum level of speed and accuracy in the typing course. Students in the advanced course might then be used to assist in preparation of the program decks for other students. This could be expecially helpful in handling the peak workloads in keypunching.

Another alternative would be to hire part-time professional operators to handle the peak workloads and to have only two full-time operators who would handle only the student workload. Assuming that 2 full-time and 2 student part-time operators could handle the workload (with the part-time persons working 1/3 time), a comparison of the costs of preparation of the student decks by the methods outlined above can now be given:

Initial creation of decks:

	Students punch own decks	2 full-time + 2 1/3-time k.p. oper.
Keypunch rental @ $60 per mo.	13 x 60 = 780	4 x 60 = 240
Full-time operators @ $3/hr. 8 hrs./day, 22 days/mo.	0	8 x 22 x 2 x 3 = 1056
Part-time operators @ $1.50/hr.		2 x 1/3 x 8 x 22 x 1.50 = 352
Totals per 1000 students per year	$780	$1648

Corrections: (assuming students make own corrections regardless of method of creation of the initial decks)

Additional keypunch rental	3 x 60 = 180	180
Totals per student per year	$960	$1828

Since this is computed for a school of 1000 this represents a cost of about $9 per student per year if the students do their own keypunching or $16 per year per student if professional keypunching service is provided. Both of these are felt to be prohibitively high, although some improvement in having the students punch their own decks might be possible if the students first took a short course in typing.

1.3 Mark-Sense* Cards

An alternative method of producing program decks is to use mark-sense cards. Several alternatives are possible:

(a) Do the initial deck creation using mark-sense cards and make the corrections on keypunches;

(b) Do the initial deck creation on keypunches and make the corrections on mark-sense cards;

(c) Do both the initial deck creation and the corrections on mark-sense cards.

Marking of well-designed mark-sense cards can be done at about the same rate as one prints block letters neatly. However, this rate of speed comes after considerable experience. This would be on the order of 15-20 characters per minute. For our assumed average program of 100 cards of 30 characters per card it would therefore require about 2-1/2 - 3 hours to initially create the deck. The average here may be misleading. At the start of the semester the students will be less experienced in using the cards, but the decks would be shorter and 1-2 hours for the creation of the deck is probably a good estimate. Late in the semester some decks will tend to be long and from 5-10 hours could be spent creating the decks. It is felt that this would have a discouraging effect on the students which may more than offset the desirable characteristics of mark-sensing: reduced preparation costs and freedom from dependence upon keypunch equipment and services.

A possible compromise between full professional keypunching support for preparation of the student decks and full mark-sensing might be to have the students mark-sense the smaller decks and provide professional keypunching support later in the semester when the decks become longer. This could probably reduce the earlier cost estimates by a factor of $2 to $8 per year per student. For the population of 100,000 this represents a cost of $0.8 million which we feel is prohibitive as well.

Specially formatted cards for mark-sensing intended to reduce the number of strokes needed may permit some reduction in the time needed to prepare the program decks. However, even with such improvements, it is felt that for programs with over 50 cards to prepare, the task of deck preparation will begin to take disproportionate amounts of student time. A further consideration is that the compiler will need modification to handle a mixture of regular and specifically formatted cards.

1.4 Low Cost Manually Operated Card Punch

A third alternative is to have students prepare their own decks using a new type hand-operated portable card punch. COMPLAN experiments in punching FORTRAN source programs using one such device indicate that a punching rate of 1 card per minute is feasible with a high degree of accuracy. This particular equipment "interprets" (i.e., prints the characters across the top of the cards at the same time the appropriate holes are punched), making it easy to do sight checking of the cards.

The portable punch we have examined is priced at about $400 per unit and is now commercially available.

The equipment is simple in design and has relatively few moving parts and therefore maintenance problems and costs are expected to be minimal. The units are expected to have a better mean time between failure than more complex electrically-operated keypunches.

In order to obtain some hard data on the economics and effectiveness of electric keypunches vs hand-marking of mark-sense cards vs keypunching with the hand-operated portable punch, we recommend that a controlled experiment be designed and conducted at the earliest feasible date, perhaps during the Fall of 1968 in conjunction with some initial training of teachers who are to teach the programming courses. Objective of the experiment would be to determine timing parameters for the preparation of source program decks by the three methods described above and thereby to determine which method(s) of source deck preparation are feasible for the proposed system.

It is our present feeling that the extreme simplicity of the manual punches will result in more effective and probably faster operation by relatively unskilled student users, as compared with more complex electric machines. We would anticipate, therefore, less wasted time and less repeated punching of the same card.

1.5 Recommendations for Student Deck Preparation

Therefore, although our calculations earlier in this Appendix showed that student-usage keypunch requirements appeared to be about 16 units per thousand students (i.e., 1000 machines for the population of 100,000 students), COMPLAN recommends that a smaller number of manual punches be obtained.

---

* The term "mark sense" has been in use for many years to denote the use of pencil marks on cards to be converted by hardware into machine-readable punched holes. The original equipment produced by IBM sensed electrical conductivity of the pencil marks. Several kinds of equipment developed in recent years use optical sensing of the reflectivity, to visible light, of the pencil marks. The equipments that seem to be potentially useful in the system described in this report are of the optical-sensing variety. We will use "mark sense" in a general way to mean any kind of hardware having the ability to sense the presence or absence of pencil-marked spots in specified locations on cards that may or may not also contain punched data.

For budgetary planning purposes, COMPLAN recommends that the pilot system be planned to include about 10 portable hand-operated keypunch units per 1000 students, to be acquired and installed on a gradual basis as usage builds up. This procedure will permit effectiveness of usage, as well as quantity requirements, to be detetmined by actual observation.

We also feel that it is important that the pilot system permit usage of mark-sense cards as well as the manually-punched cards, in order to provide for effective evaluation of the two methods of card preparation.

The mark-sense method will also permit students to prepare cards at home, in class, or during times when punching equipment may be overcrowded or otherwise unavailable.

Our assumption is that it will be worthwhile for students to wait, if necessary, for use of a card punch in order to prepare initial versions of sizable program decks (say, 100 cards or so). Correction card preparation, typically requiring creation of only one or a few cards, or preparation of entire decks for very short programs, will probably be more convenient and perhaps quicker through use of the mark-sense method.

The card readers recommended for the pilot system (cf. Appendix E) permit the reading of cards that are either pencil-marked or punched, in both cases using conventional Hollerith-coded representation of alphanumeric information. It is significant that these readers permit either marked or punched cards to be read, interspersed without restriction; and even partially-marked, partially-punched cards can be read. Such readers are about the same speed as low-priced punched-card readers (i.e., 200-250 cards per minute) and appear to be adequately reliable. They are more costly than punched-card-only readers, but the cost differential is only a few percent of system cost. Should experience indicate that hand-marked cards are preferable for student use, the saving for future system replications in cost of card punches alone would be several times the added cost of the combination readers.

1.6 Administrative Keypunching

It is recommended that the data and source programs associated with the administrative workload be prepared using a keypunch and verifier. During the first years there may not be sufficient workload in this area to justify a full-time keypunch operator. There are at least two options for schools to consider: hire a keypunch operator and cross train her in other clerical tasks, or hire an additional clerical worker and cross train her to do keypunching. The former is recommended, since that will provide a person skilled in keypunching and will make the transition to a full-time keypunch workload easier.

In regard to hiring the full-time keypunch operator, another factor should be considered. In the business oriented courses in the curriculum the programming language recommended is COBOL. COBOL is a verbose language with virtually no way to shorten the programs. It is felt that it does not lend itself well to use of mark-sense cards in preparation of the source programs. It is therefore recommended that student COBOL programs be prepared by keypunching rather than mark-sensing, and that the keypunch operator handling the administrative workload also punch the student COBOL programs.

An alternative to keypunching to meet the administrative workload requirements would be to install an optical character recognition device at the central facility and to prepare input data by means of typewriters at each school. From discussions with persons currently engaged in studies of the effectiveness of such equipment we conclude that it is presently on the frontier of the state-of-the-art and might be an appropriate change to consider in going to a replication of the pilot system. However, it is important that no unnecessary risks be taken in the development of the pilot system and use of optical character recognition would be a decided risk at this time.

2. Handling of Student Programs

2.1 Students' Own Program Decks

The large volume of program decks to be prepared and handled makes it physically impossible to provide storage space at the computer center for all of the students to leave their decks in storage when not submitted for a computer run. Consequently, the students will have to handle the decks frequently, carry them around with them, put them in lockers, bookbags, pockets, purses, etc., and the cards can be expected to show considerable wear and tear. For this reason it is important that the card reader on the system be capable of handling cards in relatively poor condition without frequent card jams, or incorrect sensing of the information contained on the cards. See Appendix E for further discussion of input/output hardware characteristics.

2.2 Operator Handling

The large number of program decks to be handled by the operator in loading and unloading the card reader make it desirable to assist the operator in identifying where one deck ends and the next begins. This will assist the operator in separating decks if they are loaded in the card reader back-to-back and in identifying complete program decks. This can be accomplished by using cards of various colors for control cards in the deck, and using a header (first) card of a unique color and opposite corner cut to the rest of the deck.

2.3 Error Form

Errors will be found to occur with sufficient frequency in processing programs that it will pay to have a short form made up on which the operator can merely check the nature of the error and return the deck to the student.

2.4 Student Program Security

The school station handling procedures for student decks and output must provide reasonable assurance that a particular student's own work will be available only to him. The procedures that need control are those by which card decks and printer output are associated with each other and are returned to the submitter.

A safe method would be to give the submitter a receipt when the deck is submitted and to require presentation of the receipt in order to obtain the output. Such a system works well in a college environment where students have more freedom of motion. However, in the high school situation this could become a bottleneck, especially at peak submission and retrieval times such as before school, between classes, at lunch hour and after school. An advantage of some form of a numerical receipting system is that the computer center can post notices periodically stating that jobs through number nnn are now ready for pick-up thereby eliminating the necessity for students to inquire as to whether their jobs are done. The bottleneck problem can be avoided by providing enough service channels for submission and pick-up of work with provision for extra help at the obvious peak periods of the day. Receipt cards can be prepared in advance for the input, so the submission is merely a matter of handing in a deck, getting a receipt number, and having the person who receives the decks place a copy of the same receipt number on the front of the deck. On the output and return side, the decks and output should be paired as the output comes from the printer and the packages must then be stored in a manner that lends itself to easy identification and retrieval. A simple approach would be to have several large tables behind a counter. The decks would then be placed on the tables in a clearly marked sequential order for easy access. The tables should be behind the counter so that student access to them without presenting their receipt would be controlled. A more ambitious project would be to build a mailbox filing system. Boys in industrial arts could build the boxes, but use of such a system would take more space and require more time to file the output in the proper boxes.

Provision would have to be made for the inevitable student who is going to lose his receipt. It is recommended that students who lose receipts have to request assistance for obtaining their deck by going to their instructor, who then must ask the computer center to look for the deck and the latter do so when time permits. If the procedure to retrieve a deck when the receipt has been lost is slow and laborious, it will tend to cause students to be more careful in retaining their receipts, which is really what one would like to achieve.

One must impress on the persons working in the production control operation of the computer center of the need for great care in handling the program decks, in matching the proper output to the program deck, and in filing the deck in its proper place on the table on in the mail boxes in order to minimize the problem of lost or misplaced decks and output. Obviously, care is also needed in seeing that the proper deck is given for the receipt submitted.

For the purpose of studying job flow from the time a student submits a job until he receives the output, it may be desirable to use an automatic stamping clock and record the time the job is submitted, the time the card deck is placed in the card reader, the time the card deck and output are matched, and the time the student picks up the deck. All four times can be stamped on the computer center copy of the receipt card and saved for later examination. Analysis of the data will give some insight into bottlenecks in the system but there is an obvious cost factor associated with collecting and analyzing the data.

In spite of all the efficiencies that are planned, there is bound to be a short, hectic period each morning when jobs are being submitted and another at the end of the school day when students have only a few mintes to pick up output and still catch the schoolbus. It should be noted that individual schools will need to do some advance planning to get students to work in production control at the computer center. Since these are the people who have access to the output, they must be selected with care to minimize the possibility of cheating problems.

2.5 Student Card Stock

Students tend to be careless with supplies they are given free. While the cost of punched cards in the computer center is quite low (about 10 cards for 1¢), wasted supplies are added costs that should be avoided. This problem might be solved by establishing that punched cards are another student-provided item of supply just as notebook paper, pencils, etc., and by making the cards available for purchase in the same manner as pencils and paper. The cards could be sold in nominal quantities such as 100 for 10¢. However, if the school elects to provide professional keypunch operators for the initial keypunching service, it probably makes sense to have the students provide their own cards only for their corrections. It should be noted that used punched cards have a salvage value. While this varies from place to place and may upon the color of the cards, it is generally in the region of \$.50 to \$1.00 per case (10,000 cards). If salvage is attempted, difficulties can be expected in getting students to place used cards in suitable containers for salvage purposes.

APPENDIX C -- On-Line Storage of Student Programs

1. Introduction

The question of whether or not student-created class-exercise programs (and, perhaps, reference file data, where used, as well) should be stored on-line in the system, or should be carried away by each student and reentered whenever used, is an almost purely economic one.

The number of active users, according to the GLC report, will approximate the total number of students enrolled. (Remark: This is justified by the assumption that system usage will be substantial in many kinds of courses and that the number of students who make no use of the system will be comparable to the number who use it for more than one course of study at a time. Consequently, there would be roughly a 1:1 relationship between enrollment and users, on the assumption that one student who uses the system for two courses constitutes two "users".)

2. Magnitude of Storage Requirements

Assume, for the purposes of this discussion that data+program storage for an average-sized student program corresponds to the object program memory space required by GLC's "Simple Language" assumptions (described by them as requiring about half the space needed for "Advanced Language" programs), viz. (cf. GLC Report, Page 6-15, Table 6.17) 11,200 characters of storage. Assume, further, that it is desired to retain in on-line file storage the current version only of half a year's programs, i.e., three programs of that size. File space required would be then:

$$100{,}00 \text{ students} \times 3 \; \frac{\text{programs}}{\text{students}} \times 11{,}200 \; \frac{\text{characters}}{\text{program}}$$

$\sim$ 3.3 billion characters.

3. Discussion of Costs

This amount of on-line file space is economically unworkable. It is several hundred percent more than recommended for the system for all other purposes, and at present-day prices would cost as much as the entire recommended central computing complex.

The practical likelihood of students using this much space is questionable, inasmuch as many users would be performing the most elementary tasks most of the time and might be unprepared, or at least reluctant, to make direct use of on-line storage of their file material. Futhermore, as noted in our observations in Section 2 of the present report, we question whether the students would quickly make as extensive use of their opportunities to use the system as is suggested by GLC. Current programs alone would exceed 1 billion characters.

Even with such reservations as to the extent to which the system would be used by the students, and even if the above assumptions are substantially reduced in magnitude, the fact remains that the potential cost of on-line storage of all student programs, at student option, appears to be too large to be feasible.

4. Recommendations

We recommend that, in general, students be required to enter their programs for each run using their own source-language program decks. Our input/output and compiling workload estimates have taken into account this operating policy for students. (Remark: The added cost of providing this service, which could be partially avoided by on-line storage, is so small as not to affect the type or quantity of equipment needed to meet the basic system requirements.)

We also recommend that a special-permission procedure be established by which teachers may authorize use of on-line file space by a selected few advanced students for specific problems and for specified periods of time only. We recommend also that means be developed for making a dollar "charge" against an (allocated) student "account" balance for the use of the file space, in order to emphasize recognition of the fact that use of system facilities costs money. This charge should be at an arbitrary level somewhat above the actual net cost of space used, and should be based upon a simple formula such as calendar time x space x an arbitrarily chosen constant.

The actual cost of having large on-line file space (unused) in the system is about 5¢ per thousand characters per month. This figure must be increased by many hundreds of percent if we are to account for system overhead for file usage, for the availability of space (i.e,. some share in cost of unused space), etc. We recommend that a flat charge of $1 per thousand characters per month be used.

APPENDIX P -- Discussion of Programming Languages for Student Use

1. Introduction

During the past several years a number of advanced programming languages, designed to avoid many of the limitations and disadvantages of the widely-used prodecural languages such as FORTRAN and COBOL, have been proposed and in some cases have been implemented. We shall discuss in this Appendix some aspects of the potential use of such languages in the proposed system.

The weaknesses of present languages that are of concern to scholars and computer scientists include:

(a) Limited generality of program structure;
(b) Limited generality of data structures that can be manipulated efficiently;
(c) Inadequate preciseness and clarity of language definition;
(d) Verbosity;
(e) Inadequate flexibility for system access and control;
(f) Inadequate extensibility of vocabulary, representation, and syntax facility.

2. ALGOL

A widely-used language that was intended to, or has since been given the facility to, abate several of these shortcomings is ALGOL, originally called (in 1958, when it was originally defined and proposed by an international committee) "International Algebraic Language", later renamed "---Algorthmic---" and finally, and more simply, ALGOL.

ALGOL has been widely used in the higher-education community and has been supported strongly by one manufacturer (Burroughs) and less strongly by several others. Three dialects of its orginal (1958) version, NELIAC, MAD, and JOVIAL, have become important in certain areas of the computing community.

3. APL

One interesting language that consists largely of an application of (clarified and made more consistent) traditional mathematical notation is APL (abbreviation for "A Programming Language", the title of the defining book by K. Iverson). APL has been implemented in a few research versions and may become more generally available within the next two years. COMPLAN personnel have observed use of, and have trivially experimented with, one version of APL that is operating on a dedicated medium-power system and is supported by powerful interactive editing capabilities in its time-sharing executive system.

It seems clear that, as suggested in the IBM report, use of APL in a version as well-supported as the one we have seen and used could reduce the amount of terminal time and system resource allocation needed by a typical student user who wishes to perform computing work of appropriate type. We feel that at this time it is not possible to quantify such time and cost savings, since appropriate implementations as part of a general-utility computing complex have not yet been devloped.

4. Use of COBOL and FORTRAN

Also, the work statements of the original GLC and IBM reports, as well as that now being' carried out by COMPLAN, require the teaching and use of conventional procedural languages FORTRAN and COBOL as well ALGOL. We feel that, in view of the wide usage and public standardization activities that have been carried forward on behalf of these three languages, it is appropriate that they be included in the facilities proposed for the system that is the subject of the present report.

As for added languages, some of these may be used through minor adaptation of existing compilers. An example is BASIC, another algorithmic language and one that emphasizes simplicity and ease of learning.

Educators considering the use of advanced or specialized languages should realize that their general applicability may be limited, inasmuch as students who learn use of these languages may attend institutions of higher learning, or may be employed by computer-using organizations, who either do not have access to software of the right type, who have adopted incompatible versions of the same languages, or who for policy reasons may require use of the publicly standardized languages.

COMPLAN beleives that the learning of computing concepts is more important than details of particular hardware, software, or language facilities that may be used in the study process. Consequently, in principle, it would seem to be immaterial what kind of languages are used so long as they permitted teaching and learning to be performed effectively.

In practice, this is true to a limited extent for academic students who are planning on higher education, although learning of a language that actually will be used will be helpful and time-saving to such a student.

For terminal (vocational training) students of data-processing practice and technology, we feel that training in the use of language(s) they will actually use is highly desirable. COBOL and FORTRAN will almost certainly be used by programming technicians who will work in business-data-processing and in scientific/technical areas, respectively. Consequently, for such students we feel that use of standard versions of COBOL and FORTRAN in the proposed facility is mandatory.

APPENDIX Q -- Cooperative Creation of Library Pro-Program by Teachers

As the use of computers begins to permeate the high school curriculum, it is obvious that it will not, and should not, penetrate each of the departments to the same depth. In some cases, the penetration will only be to the extent that the students need to be able to call out some standard programs from a library and use them to get desired results. However, it is important that this depth of penetration be encouraged, since it will permit the teacher to use realistic data in both volume and complexity.

The similarity of curricula among the participating high schools can be expected to be very high. If no attempt is made to pool efforts, teachers from many schools might work on the development of nearly identical library programs for use by by their respective students. Such a situation would be an unnecessary duplication of effort.

As a part of the effort to train the faculties of the participating high schools, some form of a cooperative group effort to develop library programs should be initiated. It may be desirable to have more than one person or small group of persons work in parallel on a given program, but the efforts should be coordinated.

Two major beneficial side effects of this cooperation in planning and producing of computer library programs will be:

(a) The faculty members of a given discipline will get together much more frequently than formerly and will inevitably get into "bull sessions" and exchange ideas for improved teaching techniques. In so doing, the younger teachers will bring with them new concepts from the teacher colleges, and the older teachers will contribute ideas tested in the classroom, and

(b) Individual programs that are planned and created cooperatively will benefit from the scrutiny of interested colleagues of the originators.

While the similarity between junior college curricula is not thought to be quite as predominant as that of the high schools, a meaningful amount of cooperative effort is also believed possible in many departments.

The major point of this appendix is that, unless some effort is made to initiate cooperative efforts, the idea may never get started.

An added benefit of creating library programs that will meet the needs of a sizable number of the schools is that this will in turn reduce the number of programs in the system library, thereby saving valuable on-line file (auxiliary memory) space for other uses.

If one is prone to argue that it is obvious that the schools and faculties will join together, numerous examples can be cited in industry and government when communications broke down, or where never initiated, and as a result a large amount of duplication of effort resulted.

Discussion points both for and against cooperative activity might be raised by citing well-known successes and failures of computer user groups such as SHARE, DUO, and CO-OP. We suggest that such group activities are not comparable to the cooperative suggested here: Most of the "user groups" are heterogeneous organizations having in common only the usage of particular computing hardware; the teacher group we propose would, on the other hand, be composed of colleagues who have closely allied interests, working circumstances, and backgrounds.

APPENDIX R -- Discussion of COBOL Language Usage for Business Oriented Students

Students studying accounting, business mathematics and other subjects related to the modern business environment will find a programming language designed to provide formatted output reports and program documentation more suitable than an algebraic compiler such as WATFOR (see Appendix S). The most popular language of this nature is COBOL, or Common Business Oriented Language, developed by a group of users comprised of representatives from government, education and industry.

Like FORTRAN, COBOL is available from all the major computer manufacturers as a standard software component, usually stipulated to meet the minimum specifications of the proposed USASI Standard as defined by COBOL Information Bulletin Number 9, January 27, 1967.

We feel that the COBOL language will be more difficult to teach than an algorithmic language, will consume more system time per student job, and will require a different approach to the creation of source program card decks.

Procedures are defined within COBOL using statements that are more natural semantically and less terse syntactically than statements written for a language "formula translator" such as FORTRAN. The PROCEDURE DIVISION OF COBOL consists of (all of the) procedure statements in a program. This part of a typical student program is relatively straightforward and simple for most student programs; and learning to prepare PROCEDUREs will probably require less classroom instruction than learning to prepare the more rigorously structured DATA DIVISION of a COBOL program.

The business student should gain a better appreciation for file handling and non-numeric processing in general than his science-oriented counterpart, concepts which currently are of prime importance to the computing community.

Requirements of the COBOL compiler in terms of program structure, keyword length and verbosity of source language program representation will generally cause more computer time to be consumed per COBOL student than would algorithmic language (e.g., FORTRAN) usage.

Although there exist COBOL compilers that are modest in main memory space requirements, we know of no COBOL compilers capable of processing programs completely in main memory, in what could reasonably be a desirable "quick-batch" mode. The compiler is usually segmented into many small pieces which then are called in from auxiliary storage as required for processing functions.

We do not believe it is practical for students to prepare COBOL source programs using mark-sense cards due to length of keywords and verbosity of the language. We recommend (see Appendix N) that COBOL source programs be keypunched and that they be processed on a 24 hour turn-around basis rather than in the "quick batch" mode.

APPENDIX S -- Discussions of a "Quick-Batch" Student Language Processor

Regardless of the system environment, whether batch mode, time-sharing, or "quick batch", the most frequently used compiler for student problems should be designed to attain the following objectives:

(1) The compiler should accept a standard language (e.g., USA Standard FORTRAN (USASI-X3.9-1966)) not a severely restricted subset, as the source language.
(2) Fast compilation speed (resulting in low complilation cost) should be of prime importance, since most programs will be recompiled several times by the novice student programmer.
(3) Comprehensive error diagnostics must be provided at both compile time and program execution time.

An illustration of past attempts to meet these criteria is provided by the University of Waterloo 's WATFOR compilor.(S1)

FORTRAN IV was selected as the source language in order to achieve maximum language compatibility with other available compiling systems (offered primarily by the computer manufacturers); and because they "...expected to use the compiler as an educational tool, FORTRAN seemed to be the most appropriate programming language to teach their students." It is noteworthy that WATFOR was slightly incompatible with, and incorporated several extensions to, its FORTRAN counterpart. The primary reason for the differences was to enable a novice programmer to keep out of difficulty when writing FORTRAN programs. An example of this desirable difference is the free format input/output facility, provided to eliminate the need for explaining the FORMAT statement at an early stage. This FORTRAN feature is generally considered as one of the more difficult concepts for the novice programmer to master.

The principal advantage of WATFOR for its intended environment -- many relatively small programs with limited I/O demands -- is that it translated source program into machine code at speeds of up to 100 statements per second on a 4 usec, 36-bit word length parallel binary computer. The University's student programs consisted of about 50 cards (main program and one or more subroutines) and were compiled (on an 8 usec machine) in two to three seconds.(S2) Execution of this type of job averaged two to three seconds, according to the amount of output requested. System overhead time was minimized by avoiding the use of peripherals during job compilation, except for input and output. Since the compiler is resident in main storage during the complete processing of a job, batches of student programs were processed as an entity in the input stream to the computer, with the consequent further reduction in operating system overhead cost.

WATFOR made use of the computer storage clock in alloting a three minute time limit for each student run.

The third criterion, the need for error diagnostics at compile and run time, was met by relating the printed error message to a serial, statement line count. The standard syntax errors are detected during the compilation of the source program, and error messages are printed which indicate the statements in which they occured. Run time diagnostics of the same format are provided by machine code generated during compilation which detects certain logical errors. In this way, inconsistencies such as undefined variables, subscript values not within the bounds declared for a subscripted variable, and the redefinition of constant parameters within a called subprogram, are detected while the program is in execution. If possible, the program-defined name of the variable in question is printed below the error message.

In deciding to concentrate on compiling speed, and job-processing efficiency in general, the designers were led to several decisions affecting performance in areas less critical to their student environment.

Since the compiler and associated run time routines were to remain in core during program execution, only relatively small programs occupying less than half of ordinarily available main storage could be accommodated.

Object programs generated by WATFOR generally executed at a rate half as fast as the manufacturer-supplied FORTRAN compiler, which did not perform run-time error checking and which optimized program execution speed rather than compilation speed.

The designers felt that, in most cases, a job presented to the system does not pass the compile-time test; and when it does, the user is quite often interested in processing only a few sets of data. Further, they believed that optimizing compilation speed made more sense within their environment since many object program inefficiencies can be traced to a poorly written source program. (S2)

WATFOR compiles source programs directly into object code and will not accept source text written in any other language than this dialect of FORTRAN IV. This feature requires external library programs to be stored in source language form and to be compiled each time they are required for program execution. Two factors affecting performance justify this choice. First, since compilation speed is optimized, the time required to relocated the object code within main storage, using the systems loader, would not compare favorable with recompilation time. Secondly, the associated run time routines mentioned earlier include the object code for all standard FORTRAN IV function, both library functions (such as SIN, COS, ATAN, etc.) and built-in functions (such as MOD, FLOAT, AMIN, etc.) which are usually all that is required by the student programmer. (S3)

Requirements for limited input/output activity meant that users could not access any peripheral devices other than the utility files predefined within WATFOR itself. This effectively prevented the user from reading permanent input files or generating output files to be saved by the system for later use.

Clearly, the dilemma facing the WATFOR designers was to provide a system which combines fast compilation with both optimization of object efficiency and systems flexibility. Their solution was to provide two systems having reasonable source language compatibility; the WATFOR compiler for the fast compilation of student jobs, and the manufacturer-supplied FORTRAN IV compiler for generation of efficient object programs for production jobs.

WATFOR has recently been implemented on a large-scale, multiprogrammed computer of the third generation, with a subsequent relaxation of some of the restrictions discussed earlier and a substantial increase in compilation speed, reaching 200 statements per second for most student programs. There are more than fifty installations using this compiler in both the student and professional environments.

We feel that experience gained with this kind of "quick-batch" software system is encouraging to the potential student user group interested in low cost, educational computer usage.

REFERENCES FOR APPENDIX S

(S1) Shantz, P. W., et. al., "WATFOR - The University of Waterloo FORTRAN IV Compiler, Communications of the ACM 10, page 41, 1967

(S2) Shantz, P. W., et. al., "A Guide to Implementing WATFOR", University of Waterloo Digital Computing Centre, Waterloo, Ontario, Canada, Revised Edition, April, 1966.

(S3) "7040 FORTRAN IV Users Guide", Computer Science Department, University of Waterloo, June, 1966.

APPENDIX T -- Recommended Configurations

	100,000 Student System	200,000 Student System
1. **HARDWARE**		
1.1 **DATA CENTRAL**		
1.1.1 Central Processor	1	1
1.1.2 Main Memory	128k* Words or 512K Bytes	128K Words or 512K Bytes
1.1.3 Bulk Core or Fast Drum	512k Words or 2m Bytes	768k Words or 3m Bytes
1.1.4 Mass Memory (Discs or Large Drums)	128m* Words or 512m Bytes	192m Words or 768m Bytes
1.1.5 Communications Processor	1	1
1.1.6 Magnetic Tape Units	4	6
1.1.7 Card Reader (1000 cpm)	1	2
1.1.8 Card Punch (500 cpm)	1	1
1.1.9 Line Printer (1200 lpm)	1	2
1.2 **REMOTE STATIONS**		
1.2.. Card Readers	57	114
1.2.2 Printers	57	114
1.2.3 Manual Punches	1000	2000
2. **PERSONNEL**		
2.1 **DATA CENTRAL**		
2.1.1 Manager	1	1
2.1.2 Shift Supervisors	2	2
2.1.3 Computer Operators	6	8
2.1.4 Tape Librarians	2	3
2.1.5 Prod. Ctl. Clerks	4	5
2.1.6 Keypunch operators	1	2
2.1.7 Clerical Workers	2	3
2.1.8 Couriers	2	3
2.2 **REMOTE STATIONS** (Personnel Stationed at Data Central)		
Maintenance Technicians	4	7

---

*k=1024 m=1,048,576

APPENDIX U -- Compiler Vs. Interpreter for Student Programs

For the proposed quick-batch student-program processing requirements, the choice between a true compiler and an interpreter is not an obvious one. Let us briefly review the design considerations in order to make clear the nature of the technical/economic decision to be made.

A compiler translates the user's source program (written, in the case of the processor described here, in Standard FORTRAN language) into an object program that often consists of a machine-language representation of the instruction steps to be executed by the computing hardware in order to accomplish the required computational procedure. Compilers may be either "monolithic" (i.e. conventional), in that they translate the entire program as a unit, or "incremental", in that they translate segments or increments of a program in order to permit some kinds of changes to be made without total recompilation A compiler must go through a translation procedure phase which requires significant use of machine time devoted entirely to program conversion, as opposed to the program execution phase which produces the desired computational results, while an interpreter does not go through a single separately determinable compilation phase.

An interpreter performs translation during execution, converting the source program one statement at a time. It might thus be thought of as an incremental compiler for which the program increment is a single statement, but which recompiles each statement every time it is executed.

The decision as to whether a given processor should be a compiler or an interpreter is usually made by considering the number of times a program is likely to be executed without change, as well as the fact that for acceptable efficiency an interpreter as well as the source program must be resident in main memory. Inasmuch as typical CUES student usage is expected to be characterized by frequent source language changes on small program decks, these considerations might seem to force an immediate decision in favor of an interpreter for the CUES quick-batch student-program processor.

On futher examination, however, COMPLAN feels that the decision is not so clear-cut and may, in fact, be a function of the detailed design of existing system software and the characteristics of the hardware. Futhermore, the question of whether the object program exists as a separate machine-language information array in main memory, after an explicit "translation phase" in which the entire program gets converted, may be almost irrelevant as regards computer CPU time and memory space economy and also as regards the cost, calendar time for preparation, predictability, and design difficulty of the processor program (i.e., the "compiler or interpreter"). In fact, the nature of the processor and of its creation will be only trivially affected by whether it turns out to be, as defined above, a true compiler or a true interpreter.

The nature of the workload, and of the basic processor performance characteristics that are required to handle it economically, cause the major design consideration of the processor to be that of an interpreter: Both source program and processor must reside in main memory at the same time and in their entirety.

The amount of machine computation that is required for source program translation by a compiler is largely determined by the amount of object program optimization or "polishing" that is performed. For example, a program (or a segment of a larger program) that will be executed many times should have considerable attention devoted to removal of redundant operations, elimination of the storing of preliminary results that will be operated upon further almost immediately, and use of simple rather than general procedures where appropiate (e.g., cascaded multiply rather than exponential operations for generation of integer powers of numbers).

This kind of object program improvement by the compiler may save substantial amounts of both CPU time and main memory space, but is costly in both time an space for operation and containment, as well as for design and creation, of the compiler itself. Some general-purpose compilers (e.g., IBM 360 FORTRAN H) provide user choice of the amount of polishing to be performed; determination of the quantitative parameters of the execution-economy tradeoff is a non-trivial programming task that may be important for large programs created by advanced users.

For the small programs that will be prepared by CUE's student users, it now appears that such user flexibility is not justified, and that relatively little optimization will be justifiable, since on the average programs will be executed less than once per compilation and since in general their execution time will be brief.

Many of the kinds of translation optimization that can be performed by compliers cannot be performed by interpreters, in that some reordering of execution step sequence is involved. Some kinds of optimization, such as creation of dummy variables to replace expressions that appear many times in a source program, can be performed by an interpreter inasmuch as they amount to improvement of the source program rather than of the object program.

Major considerations that affect the usefulness and value of the processor for the CUES student support task will not be affected by the compiler/interpreter decision. These include the language scope and fidelity to the publicly-accepted standard definition, the nature and clarity of diagnostic messages to be provided to student users to help display actual or possible program errors, and the "durability" of the processor (i.e., its freedom from loss of control resulting from user program errors or system workload exigencies).

For later replications of the CUES Pilot System, especially if a much-larger population of students is to be supported, more basic decisions may be considered. These include (a) whether read-only memory should be used for the student-program processor; (b) whether reentrant (i.e., mutiple-concurrent-user) operation of the processor is appropriate; and (c) whether consideration should be given to implementing part of the language processing function for student programs as hardware macro-operations (i.e., to providing in part "direct hardware execution of source language").

In sum, the student program processor required for efficient and economical handling of the CUES main-stream student program workload will have many of the characteristics of a large interpreter, and will be relatively little affected by the decision as to whether it will be under a formal definition, a compiler or an interpreter.

COMPLAN feels that the final choice on this matter may well be best made by individual bidders at the time this important item of CUES software is procured, inasmuch as previous work by a particular bidder might significantly affect his bid and the way in which it would be most appropriate for him to proceed in producing this program processor. COMPLAN's decision, then, subject to approval by the CUES Project Officer, is to specify this processor's characteristics in such a way that either a compiler or an interpreter may be bid, with the evaluation criteria being those of presumable overall system performance with typical student programs, bidder capabilities, price, and delivery date and predictability.